COMPUTER VERIFICATION OF THE COMPLETENESS OF A SIMULATION PROBLEM DESCRIPTION BY NATURAL LANGUAGE INTERACTION

Frederick Harold Hemphill

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

COMPUTER VERIFICATION OF THE COMPLETENESS
OF A
SIMULATION PROBLEM DESCRIPTION
BY
NATURAL LANGUAGE INTERACTION

by

Frederick Harold Hemphill, Jr.

Thesis Advisor: G. E. Heidorn

December 1971

Approved for public release; distribution unlimited.

Computer Verification of the Completeness of a Simulation Problem Description by Natural Language Interaction

by

Frederick Harold Hemphill, Jr.
Captain, United States Marine Corps
B.E., Yale University, 1962

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1971

ABSTRACT

A research project in natural language man-machine communication is currently being conducted at the Naval Postgraduate School. The system being developed, called NLPQ, is an application of a more general system, known as NLP, which consists of a rule language and the programs to compile and execute those rules. NLPQ currently consists of particular sets of NLP rules which allow a user at a time-sharing terminal to input an English text description of a queuing problem, have the computer construct an internal problem representation, and then have it produce an English text description of the problem and a GPSS simulation program to solve the problem.

The research described in this thesis produced the INTERROGATOR, a set of NLP rules for inspecting the internal problem representation to insure that it is ready to produce a GPSS program. The INTERROGATOR produces questions about missing or erroneous information, which the user may then respond to in English. It may also be used in a question-answer mode to input the entire problem.

TABLE OF CONTENTS

I. INTRODUCTION

The desirability of the computer as a tool is almost universally recognized. However, the computer remains a tool of less than universal application because of the difficulty of presenting problems to it directly by people not trained in its use. While much effort has been devoted to the development of versatile and easy-to-use programming languages, learning to write programs for a computer is still a time consuming task because of the rigidity and artificiality of those languages. The desirability of man-machine communication in natural language has been recognized for many years, and a number of research projects have addressed various aspects of the problem [1, 2].

One particular application of such a capability would be for describing a queuing problem to the computer in English text and having it produce a simulation program to solve the problem. A project which addresses this application of natural language inputs to a computer is currently being conducted at the Naval Postgraduate School [3]. The system being produced, called NLPQ, is a particular application of a more general system, known as NLP, which is being developed concurrently. NLP consists of a rule language and the programs which compile and execute those rules. In order to put this report in perspective, some background on the development of NLP and NLPQ is presented in this section. The sets of rules which are the components of NLPQ will be discussed in more detail in the next section.

A. BACKGROUND

The problem of translating a problem described in a natural language such as English into some standard representation which can be processed by the computer can be approached using one of several language theories as a basis. The best known of these theories is that of transformational grammar by Noam Chomsky [4]. Another important theory is that of stratificational grammar by Sydney Lamb [5]. It is on the latter theory that the development of NLP and NLPQ is based.

The immediate goal of NLPQ was to produce a system capable of accepting an English description of a simple queuing problem and producing a GPSS simulation program to solve the problem. The long range objective was the development of a general system capable of handling a variety of input and output languages to enhance man-machine communication. Initially, the basic components of the NLP system were developed: a set of FORTRAN programs which perform the functions of a monitor for the system, and a rule language whose statements are compiled and executed by the FORTRAN program. This system was designed to run on the IBM 360/67-CP/CMS time-sharing system at the Naval Postgraduate School. With this basic system established, work was begun on developing NLPQ, a specific set of rules in the NLP system which would accept an English description of a queuing problem and produce a GPSS program. The initial problem was to decide on the format of the Internal Problem Representation, or IPR. Once the IPR format was established, GPSS encoding rules, written in the NLP rule language, were written to convert an IPR into a GPSS program. The research performed in accomplishing

these two steps is reported in GES: A Data-Structure-to-GPSS Encoding System, by Richard C. Hansen [6].

The next step was to develop English encloding rules so that the computer could produce an English text description of the internal representation of a problem. The GPSS rules were also improved so that the GPSS program would be more readable. This research is reported on by Robert T. McGee in The Translation of Data Structure Representations of Simple Queuing Problems into GPSS Programs and English Text [7]. The English encoding rules developed were general in nature and hence applicable to areas other than just the description of the IPR, as will be seen in the course of this report.

More recent work has resulted in a set of decoding rules which allow the user to describe his problem to the computer in English. These rules process the English text to produce the IPR. This major step has made real man-machine interaction possible. However, now the user, in describing a problem, might omit essential information or include erroneous information which, in turn, would result in an incomplete or incorrect GPSS program. This situation indicated the need for a means of detecting missing or erroneous information in the IPR and requesting that the user supply or correct the information required.

B. THESIS OBJECTIVE

The research objective of this thesis, then, was to develop an interactive INTERROGATOR for NLPQ which would inspect the internal problem representation to insure that it is ready to produce a GPSS program. The availability of the English encoding and decoding rules allowed the construction of the INTERROGATOR entirely in the NLP

rule language by making use of those rule sets for interaction with the user. Furthermore, the input and output flexibility allowed by those two sets of rules made it possible to design the INTERROGATOR rule set so that the user could input his entire problem in a question-answer mode. This question-answer capability of the INTERROGATOR was a useful by-product of the research.

C. ORGANIZATION OF THE THESIS

Section II of this report provides background information on NLP and NLPQ. Section III presents an annotated sample terminal session to provide an illustration of INTERROGATOR's capabilities. Section IV discusses the concepts involved in the INTERROGATOR, its integration into NLPQ, and the rules. Finally, Section V presents conclusions and recommendations.

II. DESCRIPTION OF NLP AND NLPQ

Since the INTERROGATOR is integrated with, and relies on, the other components of NLPQ, a description of those components will be presented. The basic concepts involved will be discussed first, followed by discussions of the IPR, the decoding rules, a set of rules known as the MASSAGER, the encoding rules, the capability of NLPQ to recognize certain key words for control purposes, and finally, modifications made to these sets of rules concurrently with the development of the INTERROGATOR.

A. BASIC CONCEPTS

As stated in the introduction, the basic machinery of the NLPQ system is known as NLP and consists of a set of FORTRAN programs and a rule language. The FORTRAN programs include a monitor and a set of subroutines for performing such functions as compiling rules written in the system's rule language, executing operations according to those rules, and performing some input-output operations essential to the system. Understanding of the FORTRAN portions of NLP is not essential for understanding this report.

A rule written in the rule language of NLP consists of two parts separated by a special symbol (--->). The left part is a description of some state of a portion of the system, which, if it exists will cause the rule to be invoked. This, in turn, will result in changes to the system's state according to the description on the right of the symbol. In the processing of NLPQ rules, various basic elements, known as records, establish the state of the NLPQ system. These records vary

in their degree of permanence from those that are used to represent the rules, objects, and other basics of the system, through those that are created to represent the problem being input to the system, to those which create or modify other records and then disappear. These last records, known as segments, are the type whose state is most frequently tested by the NLPQ rules.

The various components of NLPQ are specified by sets of these rules. These components include the set of rules for decoding a string of input text and forming an internal representation of the problem, those rules which massage the IPR to remove information necessary only during decoding and to set certain default conditions, and the rule sets which encode the IPR in English or GPSS. Each of these sets of rules will be considered below, but first, the internal problem representation will be discussed for it is central to the NLPQ system.

B. INTERNAL PROBLEM REPRESENTATION

The data structure used by NLPQ for the IPR is an entity-attribute-value structure. That is, the basic elements of the structure represent entities, such as physical objects or actions, and these entities have attributes, such as color or duration, which in turn have values. For example, the input sentence, "cars are serviced at the pump for 10 minutes," would result in the following element in the IPR:

Attribute	Value
SUP	Service
GOAL	Car
DURATION	10 minutes
LOCATION	at the pump

This same element, known as a record, would have also resulted from the following sequence of sentences: "Cars are serviced." "The time for servicing is 10 minutes." "They are serviced at the pump."

The record shown above is somewhat simplified, since the GOAL attribute of the record for "service" would not actually contain the word "car" but rather a pointer to another record for "car" containing its attributes. The values of other attributes can be considerably more complex than shown in this example. In addition, attributes whose values are binary (e.g., yes or no, on or off) are called indicators and are treated specially in NLP to conserve space.

An example of a queuing problem which results in a representative IPR is stated in Figure 1. The IPR for this problem is shown in Figure 2. It should be noted in Figure 2 that access to almost any record in the IPR, except the special one known as MEMORY, can be gained by following the proper pointers starting from the record identified as 'ACTNLIST', or action list. Each of the records pointed to from the 'ACTNLIST' represents some action, as indicated by the value of its SUP attribute, and each of these actions consolidates some subset of the information in the IPR through its attributes. This important characteristic of the IPR allows the INTERROGATOR to conduct the major portion of its investigation by starting with each of the actions on the action list. Similar lists exist, but are not shown in Figure 2, for mobile entities ('MOBLIST'), stationary entities ('STALIST'), and several other entities in the system. The two lists mentioned specifically are also used by the INTERROGATOR.

The MEMORY, or MEM, record in the IPR is used to contain or point to certain basic problem information, such as the length of time

Vehicles arrive at a station.

The station has just one pump.

A vehicle will leave the station immediately after arriving if the length of the line at the pump is not less than two.

Otherwise, it is serviced there; then it leaves.

Service times are exponential, with a mean of 5 minutes for cars and 9 minutes for trucks.

Three quarters of the vehicles are cars and one fourth of them are trucks.

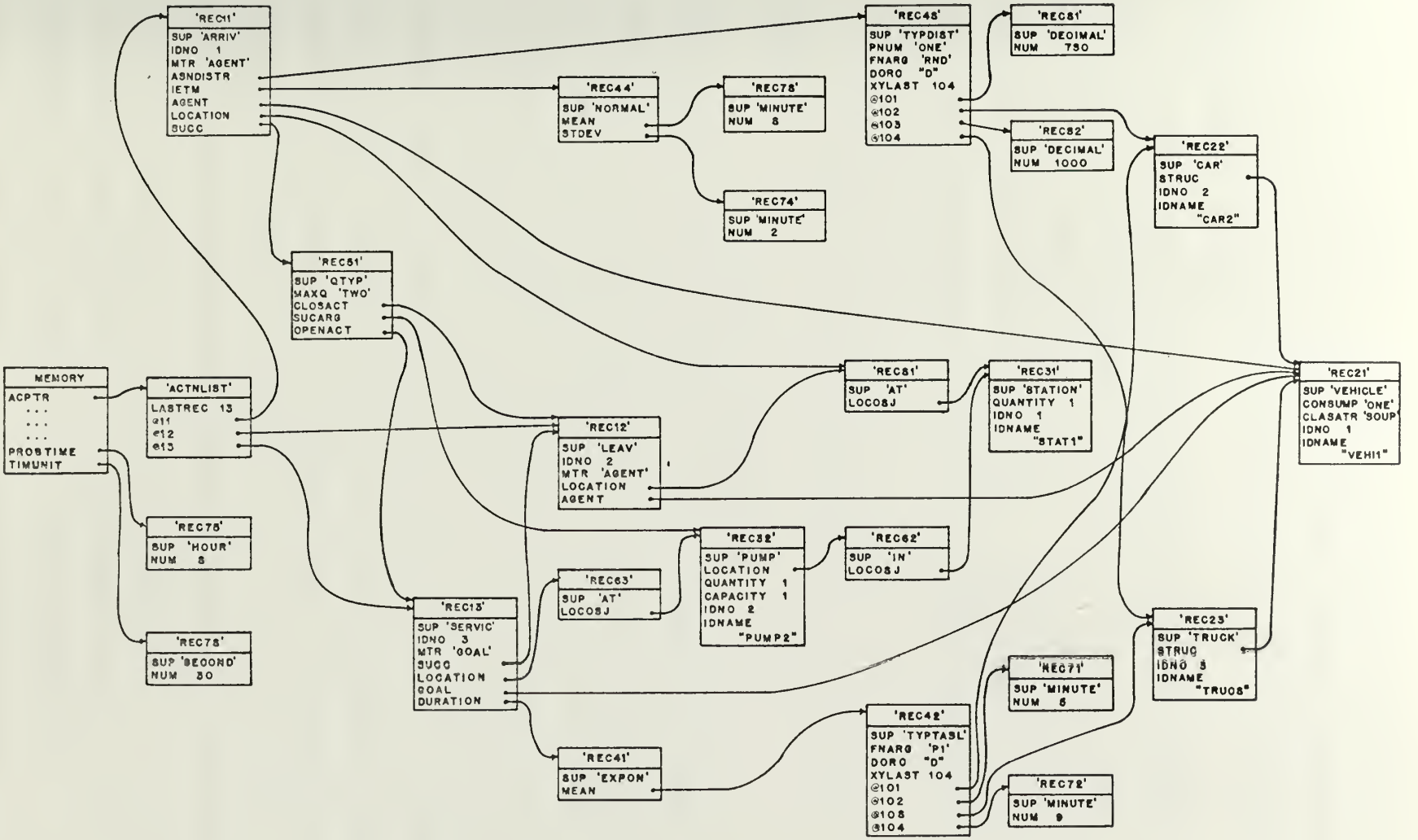
Arrivals are normally distributed with a mean of eight minutes and a standard deviation of two minutes.

The simulation run time desired is eight hours.

The basic time unit to be used in the model is 30 seconds.

A Sample Queuing Problem

Figure 1.



The IPR for the problem of Figure 1.

Figure 2.

for which the simulation is to be run, and to store temporary information which must be accessible to different components of the system during processing. Many examples of its use can be seen in the INTERROGATOR rules.

One special type of record important to an understanding of the IPR and the various components of NLPQ is the named record. Named records have several purposes. At one level, some of them define the natural language vocabulary the system can recognize. At another level, the concepts represented by words in the natural language are related to other concepts.

For example, the named record definition for car appears as follows:

CAR ('VEHICLE')

The existence of this definition allows the system to recognize the word "car" in a string of input text. In the same named record definition, the concept represented by that word is defined in part by some of the attributes and their values as listed in parentheses. In this case, the only attribute is SUP and its value is 'VEHICLE'. (The only attribute whose value is recognized without mention of the attribute name is SUP. It may be referred to by a string of eight or fewer characters enclosed in single quotation marks.) The SUP attribute is based on the notion of superset or class, and it is through the set relationships established by the SUP attributes of records in the system that conceptual relationships are established. For instance, the named record definition for TRUCK shows it to have a SUP of 'VEHICLE'. In turn, VEHICLE has a SUP of 'MOBENTY', or mobile entity, as do SHIP and PERSON. A pictorial representation of their

relationships, expanded to include CUSTOMER and MAN as members of the set PERSON, might take the form of a tree structure as shown in Figure 3.

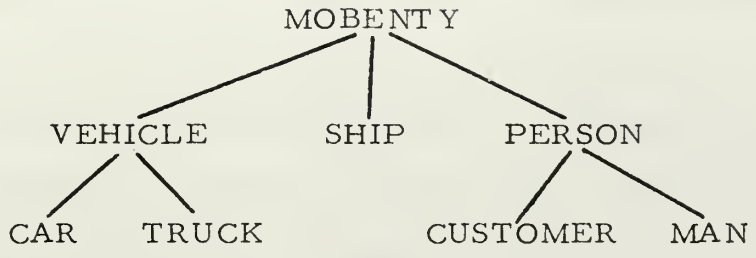


Figure 3.

Thus, the concepts represented by the words CAR and TRUCK are related, for they are both members of the set VEHICLE. Similarly, the concepts CAR and PERSON are related in that they are both mobile entities. It is important to note that while the names attached to these concepts are English or English-based, the relationships are abstract and do not depend on the natural language recognized by the system.

This relationship structure is very useful in NLPQ, for it allows a single general rule to say something about mobile entities as a class, for example, rather than requiring five separate rules. Furthermore, the capability of traversing up the SUP chain, which is provided in the system, precludes having to store extra information in each named record. Thus, the fact that a car is a mobile entity can be ascertained by traversing the SUP chain from CAR to VEHICLE and finding that VEHICLE is in the set MOBENTY. This technique is used frequently in the INTERROGATOR rules.

C. DECODING RULES

The decoding rules specify how input text is to be processed, with the end product of such processing being the IPR. The procedure generally occurs in several steps, each of which is less dependent on the language used for the input. This is a result of the stratificational grammar theory of Sydney Lamb, as mentioned before. As applied in NLPQ, the theory identifies three levels of language structure by the names morphology, lexology, and semology. While the distinctions between these strata are sometimes blurred, it can be said that morphological elements of NLPQ are closely related to the structure of some particular language, while semological elements are more concerned with what relationships the input text conveys. The lexology represents a middle ground.

For example, one of the rules in the English decoding morphology is

```
VERBS(ING) I N G    -->
            VERBP(SUP(VERBS), PRESPART)
```

This rule says that if a verb stem (VERBS) segment tested by this rule has its ING indicator set and is followed by segments representing an I, an N, and a G, then a verb part (VERBP) segment should be created which has the same SUP as the VERBS segment and which has its PRESPART (present participle) indicator set. Thus, in scanning the input character string, if the word "arriving" is encountered, this rule would be executed in the following way. After scanning the A, R, R, I, and V, a table lookup procedure would identify this string as corresponding to the name record definition for 'ARRIV':

```
ARRIV    ('EVENT', E, ES, ING, ED, ER)
```

and through the application of some decoding rules, a VERBS segment would be produced with a SUP of 'ARRIV' and its ING indicator set. Since it is followed in the text stream by the letters I, N, and G, the conditions on the left of the rule are met and it can be executed. The verb part created is of the form,

VERBP('ARRIV', PRESPART)

This is a simplification of what actually occurs during the decoding process but is representative of the essential actions which take place.

Clearly, such a rule in the morphology is specifically for English. By contrast, however, the rules in the semology are very much removed from the input language. For example, the rule,

ACTSENT(CONDITN) -->
ACTSENT(CONDITN(MEM)=CONDITN, -CONDITN)

says that an action sentence (ACTSENT) with a condition attribute (CONDITN) creates a new action sentence which is a copy of the old one, sets the CONDITN attribute of MEMORY equal to the value of the CONDITN attribute of the ACTSENT, and then erases the CONDITN attribute of the ACTSENT. Although the acronyms are based on English words, at this point, the action sentence represented by the ACTSENT segment could as easily have been derived from French as from English, for characteristics of the concepts represented by the text string are being discussed rather than the words or the relative position of the parts of speech.

The lexology represents the middle ground. While words at this level are treated as parts of speech so that they are somewhat removed from the natural language, their context may still be closely related to the language. For example, a lexological rule which tested

the relationship of adjectives and nouns might have to be rewritten if Spanish were the natural language being considered rather than English, since in Spanish, adjectives generally follow the nouns they modify rather than precede them as in English. Thus, changing languages involves writing new decoding rules at the morphological level, and some at the lexological level, but generally none at the semological level.

D. THE MASSAGER

The decoding rules construct the IPR as information is supplied by the user. Since this information is provided, and acted on, sentence by sentence, the system has no way of knowing what information may be supplied in future sentences. Therefore, certain assumptions made by the system about routine values are delayed until the user has indicated that these default values may be supplied. The purposes of the rules of the MASSAGER are to provide these assumptions about the IPR, to consolidate information which may be provided in more than one sentence, and to remove certain attributes created just for decoding purposes. The assumptions made by the MASSAGER are of the sort normally made by an individual who is knowledgable about the problem being described to him. For example, if a mobile entity, such as CAR, has been specified but no mention is made of how many units of storage capacity the mobile entity will occupy if it enters some stationary entity designated as a storage with some maximum capacity, the system assumes a consumption of one. Such assumptions are expected to be non-controversial; if there is any doubt about the value to be provided, the test for the value and the related question

are included in the INTERROGATOR rules. And, of course, the assumed values can be overridden by reentering the decoding mode and providing new values.

E. ENCODING RULES

The two sets of encoding rules currently available in NLPQ are for English and GPSS. The encoding rules essentially perform the inverse function of the decoding rules: they convert the IPR into an "equivalent" representation in some language. But in addition, they may perform other functions. For example, the English encoding rules are used extensively by the INTERROGATOR to produce descriptions of parts of the problem represented in the IPR and the related questions. The English encoding rules are also used by the GPSS encoding rules to produce comment cards interspersed throughout the GPSS program.

The basic format and stratification of the encoding rules is the same as that of the decoding rules, with some minor differences in the way they are processed. Since an understanding of how rules are processed in NLP will be helpful in understanding the discussion of the INTERROGATOR rules which follows in Section IV, the encoding rule processing procedure will be presented briefly here.

As mentioned above, the transient records known as segments are the principle elements of the system examined by rules in the rule language. For each type of segment with a different name, there is a permanent record known as a SEGMENT TYPE record with a NAME attribute containing the name of that type of segment and with another attribute pointing to the list of the rules which have segments with that name on the left of the arrow. The basic format of these

encoding rules is

```
SEGMENT TYPE(condition 1, condition 2, ...)  -->
        SEGMENT TYPE(action 1, action 2, ...)
```

There may be more than one SEGMENT TYPE on the right side of the rule. The required conditions might be the presence or absence of an attribute or indicator in the segment, satisfaction of some relationship between two values, or similar tests on attributes or indicators of some other record, such as MEMORY or an original IPR record. The actions performed might be the copying of all or part of some record into a new segment, addition or deletion of attributes of a record, or the setting of indicators. An additional feature of the rules is the capability of including a condition on the right of the arrow symbol. For example, in the format above, another condition could be specified following "action 2," with further actions to the right of the condition which are executed only if the condition is met. Thus, a hypothetical segment called TYPE1 might create a TYPE2 segment according to the following rule:

```
TYPE1(COND1, COND2)  -->
        TYPE2(ATTRA=1, ATTRB=2, ATTRC(MEM).EQ.0,
              ATTRB=3)
```

The TYPE2 segment created by this rule would have its ATTRB attribute initially set to 2. If the ATTRC attribute of MEMORY has a value equal to zero, then the test on the right would be passed, and ATTRB would be set to 3 instead.

The processing of rules is conducted through the use of two stacks, one for SEGMENT TYPES, with its pointer, STP, and one for segments, with its pointer, SP. The stacks are push-down, or last-in-first-out, stacks. However, as segments are created on the right of a rule,

they are placed on the stack in inverse order, so that the first one created by that rule will be the first one processed. When a segment is placed on its stack, its corresponding SEGMENT TYPE is placed on the SEGMENT TYPE stack. When a rule has finished creating new segments and adding them to the stack, the segment pointed to by SP is compared against the list of rules of the SEGMENT TYPE pointed to by STP until the conditions of one of those rules match those of the segment, and that rule is executed. The stacks grow and shrink, accomplishing sequences of tasks until the stacks are empty.

F. KEY WORDS

To enhance the user's control of the NLPQ system, the concept of controlling certain functions of the system by mentioning certain key words while in the decoding mode was introduced. The context of these key words need not be words known to the system, for processing will continue beyond unknown words until a key word is recognized or the end of the sentence is reached. If a key word is found in such a sentence, the function it requests will be executed. Otherwise, the user will be told that his sentence is not understood. Examples of key words which relate to the INTERROGATOR follow.

If the words "English," "state," or "describe" are used in a sentence, the system will call the MASSAGER, followed by the INTERROGATOR, with control returning to the decoding rules until the problem is completely stated, followed by the English encoding rules. "GPSS" or "program" produces the same sequence with the GPSS encoding rules instead of the English rules. The words "ask," "question," "prompt," or "inquiry" result in the MASSAGER, followed by the INTERROGATOR in the normal question-answer mode, followed by the statement, THE

PROBLEM IS COMPLETE, when all information has been supplied. One of the words "ok," "okay," "complete," "done," "check," "explain," "clarify," "what," or "which" used at any point in the decoding process will result in calls to the MASSAGER and INTERROGATOR followed by the same completion message mentioned above. If one of these words is used just after the decoding process is begun, the result is an expanded question-answer mode. The difference between this expanded mode and the normal mode is that more complete descriptions of certain portions of the problem are provided before a question is asked in the expanded mode. Such detail is necessary if the INTERROGATOR is entered at some intermediate point, but is not normally required for the question-answer mode. The differences between the expanded and normal modes will be pointed out in the following section. One other key word recognized is "stop." If the user desires to halt the INTERROGATOR and return to the decoding mode, that word will produce the desired result.

G. MODIFICATION OF RULES

Concurrently with the development of the INTERROGATOR, new capabilities were added to the decoding and English encoding rules. The decoding rules were modified to allow phrases to be used in response to questions asked by the INTERROGATOR, to accept a semicolon as a punctuation mark in the input text, and to handle various other situations introduced by the INTERROGATOR or previously anticipated as desirable. Considerable modification of the English encoding rules was made to allow the INTERROGATOR to ask its questions in an efficient manner. Some of these changes will be mentioned

when the INTERROGATOR rules are presented. A detailed discussion of these changes will be included in a later report.

III. A SAMPLE INTERROGATOR SESSION

Now that the basic components of NLPQ have been presented, the remainder of this report will be devoted to a discussion of the INTERROGATOR and its integration into NLPQ. In order to illustrate the capabilities of the INTERROGATOR an example terminal session will be presented in this section. Some discussion will be interjected in this sample problem, but the major presentation of the detailed operation of INTERROGATOR will be deferred to the next section.

In the sample session that follows, user inputs are in lower case letters while the computer's responses are in capitals. Several forms of response to INTERROGATOR's questions are allowed. Appendix A contains a listing of the basic types of questions and typical responses. Appendix B is the actual terminal session from which this example was prepared. Another example appears in Appendix C.

Arrive.

Check the IPR, please.

WHAT ARRIVES?

Cars arrive.

(Note that all that is required to start the INTERROGATOR is a single action verb. Once the key word "check" has been mentioned, the computer will ask questions until told to "stop" or until the problem is complete.)

WHERE DO THE CARS ARRIVE?

At the station.

(Each response must end in a period.)

HOW OFTEN DO THE CARS ARRIVE AT THE STATION?

The time between arrivals is exponentially distributed.

(If the action being considered is one that involves a duration rather than an interevent time, a different question is asked. See the example for "service" below.)

THE TIME BETWEEN ARRIVALS OF THE CARS AT THE STATION IS EXPONENTIALLY DISTRIBUTED. WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION?

7 pounds.

THE TIME BETWEEN ARRIVALS OF THE CARS AT THE STATION CAN NOT BE EXPONENTIALLY DISTRIBUTED, WITH A MEAN OF 7 POUNDS, BECAUSE THE UNITS ARE WRONG. WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION?

7 minutes.

(Quantities are inspected for the proper units. Here, a time is required.)

AFTER ARRIVING AT THE STATION, WHAT DO THE CARS DO?

If the length of the line is less than 4, the cars are serviced.

AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE IS LESS THAN 4, THE CAR WILL BE SERVICED. WHERE IS THE LINE?

At a pump.

AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE AT THE PUMP IS LESS THAN 4, THE CAR WILL BE SERVICED. OTHERWISE, WHAT DO THE CARS DO?

They unload at the dock.

WHERE ARE THE CARS SERVICED?

At the pump.

(After resolving the alternative actions to follow "arrive," the inspection of that action is complete. Now the first named of the two alternative actions is considered in more detail.)

HOW LONG ARE THE CARS SERVICED AT THE PUMP?

Service times are exponential.

THE TIME FOR THE CARS TO BE SERVICED AT THE PUMP IS EXPONENTIALLY DISTRIBUTED. WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION?

Normally distributed, with a mean of 3 minutes.

(The system allows such complex specification of the parameters of distributions to any depth. Furthermore, the check on units is carried down to all levels.)

THE TIME FOR THE CARS TO BE SERVICED AT THE PUMP IS EXPONENTIALLY DISTRIBUTED, WITH A MEAN WHICH IS NORMALLY DISTRIBUTED, WITH A MEAN OF 3 MINUTES. WHAT IS THE STANDARD DEVIATION OF THE NORMAL DISTRIBUTION?

2 minutes.

AFTER BEING SERVICED AT THE PUMP, WHAT DO THE CARS DO?

If the dock is not available, they leave the station.

AFTER BEING SERVICED AT THE PUMP, IF THE DOCK IS BUSY, THE CAR WILL LEAVE THE STATION. OTHERWISE, WHAT DO THE CARS DO?

Otherwise, they unload at the dock.

(This sentence could have been included with the previous response by using a semicolon in place of the period after "station." In that case, this question would not have been asked.)

HOW LONG DO THE CARS UNLOAD AT THE DOCK?

Until a truck is available.

THE CARS CAN NOT UNLOAD AT THE DOCK UNTIL THE TRUCKS ARE AVAILABLE, BECAUSE THE CONDITIONAL DURATION ENTITY MUST BE STATIONARY. HOW LONG DO THE CARS UNLOAD AT THE DOCK?

Until a window is available.

(The requirement that a conditional duration must be based on a stationary entity is a restriction of the current version of NLPQ.)

AFTER UNLOADING AT THE DOCK, WHAT DO THE CARS DO?

They load at the window.

HOW LONG DO THE CARS LOAD AT THE WINDOW?

For 3 minutes.

AFTER LOADING AT THE WINDOW, WHAT DO THE CARS DO?

20 percent of the cars leave the station, and 60 percent unload at the depot.

THE FOLLOWING PERCENTAGES DO NOT TOTAL 100: AFTER LOADING AT THE WINDOW, 20 PERCENT OF THE CARS LEAVE THE STATION, AND 60 PERCENT UNLOAD AT THE DEPOT. AFTER LOADING AT THE WINDOW, WHAT DO THE CARS DO?

40 percent of the cars leave the station, and 60 percent unload at the depot.

HOW LONG DO THE CARS UNLOAD AT THE DEPOT?

6 minutes.

AFTER UNLOADING AT THE DEPOT, WHAT DO THE CARS DO?

They leave the station.

HOW LONG SHALL THE SIMULATION BE RUN?

For 100 minutes.

WHAT IS THE BASIC TIME UNIT TO BE USED IN THE MODEL?

30 seconds.

THE PROBLEM STATEMENT IS COMPLETE.

State the problem in English.

THE CARS ARRIVE AT THE STATION. THE TIME BETWEEN ARRIVALS OF THE CARS AT THE STATION IS EXPONENTIALLY DISTRIBUTED, WITH A MEAN OF 7 MINUTES. AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE AT THE PUMP IS LESS THAN 4, THE CAR WILL BE SERVICED AT THE PUMP. OTHERWISE, THE CAR WILL UNLOAD AT THE DOCK UNTIL THE WINDOW IS AVAILABLE. THE TIME FOR THE CARS TO BE SERVICED AT THE PUMP IS EXPONENTIALLY DISTRIBUTED, WITH A MEAN WHICH IS NORMALLY DISTRIBUTED, WITH A MEAN OF 3 MINUTES AND A STANDARD DEVIATION OF 2 MINUTES. AFTER BEING SERVICED AT THE PUMP, IF THE DOCK IS BUSY, THE CAR WILL LEAVE THE STATION. OTHERWISE, THE CAR WILL UNLOAD AT THE DOCK UNTIL THE WINDOW IS AVAILABLE. AFTER UNLOADING AT THE DOCK, THE CARS LOAD AT THE WINDOW FOR 3 MINUTES. AFTER LOADING AT THE WINDOW, 40 PERCENT OF THE CARS LEAVE THE STATION, AND THE REST UNLOAD AT THE DEPOT FOR 6 MINUTES. AFTER UNLOADING AT THE DEPOT, THE CARS LEAVE THE STATION.

THE SIMULATION IS TO BE RUN FOR 100 MINUTES, USING A BASIC TIME UNIT OF 30 SECONDS.

(If a GPSS program had also been requested, it would follow the English text.)

This problem is obviously artificial, but it does serve to demonstrate the capabilities of the INTERROGATOR. In the next section, the inner workings of the INTERROGATOR will be discussed in detail.

IV. THE INTERROGATOR

The example given in the previous section illustrates some of the flexibility of the INTERROGATOR in operation. This section will present a detailed examination of the construction of the INTERROGATOR. The underlying concepts will be presented first, followed by a discussion of the integration of the INTERROGATOR into NLPQ. Finally, the INTERROGATOR rules will be discussed in detail.

A. BASIC CONCEPTS

As was mentioned in the discussion of the IPR shown in Figure 2, access to a majority of the IPR records can be had through the action records, which in turn are listed in the 'ACTNLIST' record. When the INTERROGATOR is called, it traverses the 'ACTNLIST' and creates a copy of each action record in the IPR. It is this copy which is the vehicle for the testing conducted by the INTERROGATOR. Each action is inspected by the INTERROGATOR to insure that it has all the attributes required for processing by the GPSS encoding rules. The action records are generally checked in the order in which the user mentions them, except that actions mentioned in connection with a complex successor to another action are checked at that time, but only to insure that they have a mobile entity to serve as the subject; they receive their detailed inspection later. If a question is asked by the INTERROGATOR in the course of its inspection, all further checking ceases until the user has supplied a response. Any segments which have been created for further tests go to NULL, and control returns to the decoding rules to wait for the user's reply. After the

user replies, the INTERROGATOR is invoked again, and it starts its inspection of the IPR from the beginning. However, reinvestigation of some portions of the IPR is avoided through the use of the CHECKED attribute, as will be discussed below. This procedure will become clearer when the rules are considered.

The successor of each action is important; in fact, one of the attributes required of each action except "leave" is SUCC, or successor, which indicates what occurs immediately after that action. When the INTERROGATOR has checked all actions on the 'ACTNLIST', it has insured that each has a SUCC attribute and it has also marked every action which is the successor of some other action by setting the REACHED indicator. This occurs even when the successor is complex and involves several actions, as will be seen when the rules are discussed. A second pass down the action list will identify any action which cannot be reached from some other action by its lack of a REACHED indicator and ask a question to rectify the situation. Thus, INTERROGATOR provides basic assurance that the IPR can be used to produce a GPSS program that is complete as far as its actions are concerned. This global check on the completeness of the IPR is an important feature of the INTERROGATOR.

B. INTEGRATION INTO NLPQ

INTERROGATOR was written to augment the interface between the user and the Internal Problem Representation. It acts as an intelligent listener, asking questions only when information is found to be either missing or in error. As such, the INTERROGATOR fits into the NLPQ process after decoding and massaging have taken place

but before encoding occurs. While this conceptual integration of the INTERROGATOR into NLPQ is simple, the actual integration of the INTERROGATOR rules into NLPQ is somewhat more involved. The INTERROGATOR rules inspect the IPR and set up the form of the question to be asked, but the English encoding rules are called upon to produce the question, and then the decoding rules process the user's response and insert the information into the IPR. Next, the massaging rules are called upon to process the new IPR before control is given back to the INTERROGATOR.

The control module of NLP is sufficiently flexible to allow several options in the use of the INTERROGATOR. As mentioned before, it can be used as a question-answer system to input the information piece by piece. Or, the problem can be explained in what the user hopes will be its entirety, and then the INTERROGATOR can be called to conduct its checks. And of course, the INTERROGATOR can be called at any intermediate point in the processing of a problem.

When the INTERROGATOR is invoked, it enters the question-answer mode in one of its two forms. If a question is asked, certain basic actions occur which relate to the shift of control among the INTERROGATOR, the English encoding rules, and the decoding rules. These actions will be described briefly here in general terms and again in the discussion of the rules. The most universal of these basic actions is the setting of the QUESTSW indicator of MEMORY. When this indicator is set, it has the effect of stopping further investigation of the IPR, since any segments remaining on the stack will execute rules which go to NULL because of QUESTSW(MEM) being set.

Two attributes of MEMORY which are generally set for use by the decoding rules in inserting information from the user's response in the proper place in the IPR are the CENTY and ATTRIB attributes. CENTY(MEM) is set to point to the IPR record which is to be changed by the user's response, while ATTRIB(MEM) contains the name of the attribute to be changed in, or added to, that record. In order to set these two attributes properly, four attributes are used in the segments created by the INTERROGATOR. These are CENTY (change entity), AENTY (action entity), ATTRIB (attribute), and ATTRIB1 (attribute one).

CENTY points to an existing record in the IPR which is under consideration for altering by changing or adding an attribute. As a portion of the IPR is traversed starting at some action record, CENTY will be changed to point to a new record only when that record is found to be tentatively satisfactory but in need of further investigation of its attributes. As the investigation moves further from the original action record, a tie is maintained with that action record through the AENTY attribute. AENTY points to the original record for two reasons: first, because the English encoding rules will require information frequently from that record, and second, because even though some intervening records may have been found tentatively acceptable, further investigation may indicate that respecification of a top level attribute is the surest way of correcting an error in the IPR.

ATTRIB is set to the name of the action record attribute currently under investigation. As investigation of records subordinate to that attribute of the action record occurs, ATTRIB1 is used to contain the name of an attribute being checked in one of these subordinate records.

Both of these are necessary since the English encoding rules frequently require an ATTRIB attribute containing the name of one of several attributes of action records to produce the proper sentence, even when that sentence is initiated below the action record level, and yet, regardless of level, it is generally necessary to know the name of the attribute under investigation so that ATTRIB(MEM) can be set. The ATTRIB attribute also plays a role in how certain questions are asked; this will be discussed when the INTSENT1 and INTSENT3 rules are explained. Examples of all of these cases can be seen in the discussion of the rules.

As can be seen in Appendix A, considerable variety is allowed in the responses the user can make to questions posed by the INTERROGATOR. This is a result of the generality of the decoding rules of NLPQ.

C. THE INTERROGATOR RULES

In this section the INTERROGATOR rules will be discussed in detail. These rules are listed in Appendix C. The numbers in the listing have been added for ease of reference here; they are not actually coded in the program. Reference to Appendix D may also be helpful since the various segment types are presented in summary form there.

1. Organizing Rules

The top-level call to INTERROGATOR is guaranteed to be satisfied since there is only one rule with INTERROGATOR on the left, and it has no conditions to be met. It creates six segments to be processed further to organize the processing conducted by the

INTERROGATOR. SETAGL insures that the MOBENATR attribute of an action record is properly set if the AGENT or GOAL already exists. Its rules appear in the English encoding rules. Each of the four RECLISTI segments makes a copy of one of the three lists--action list, mobile entity list, and stationary entity list--which form an index of the major components of the IPR, and adds some additional attributes to aid in processing. Finally, a MEMREC1 segment is created to check two attributes of MEMORY.

Each RECLISTI segment is processed by Rules 2 and 3 to create several RECI's, each of which is a copy of one of the entries on the list copied by the RECLISTI, with several other attributes copied or created at the same time. The most important of these is CENTY, or change entity, an attribute which points to the original record in the IPR of which the RECI is a copy. The CENTY attribute of a segment is used throughout the INTERROGATOR to point to the original record which is to be changed.

Each RECI is converted by one of five rules into one of the operational segment types of the INTERROGATOR. Those RECI which are copies of action records in the IPR and which have a K attribute with a value of 3, form ACTNREC3's which are copies of the RECI's. Note that those RECI's with K=3 are processed only after all of those which are copies of action records but which do not have a K attribute. This provides the second pass down the action list mentioned above in discussing the checking of successors to each action and the setting of the REACHED indicator on all actions to which flow can occur. A brief look ahead to the ACTNREC3 rules shows that the only ACTNREC3's which result in a question are those

without a REACHED indicator whose SUP is not 'ARRIV' or 'ENTER'. The question asked in such a case is one of the form, BEFORE BEING SERVICED, WHAT DO THE CARS DO?

Once an action record has been inspected and found to be complete, its CHECKED indicator is set so that no further checks are conducted, as provided by Rule 5. This saves time, particularly when the action list is long and actions further down the list are being checked.

RECI's with their LIST attribute equal to 'ACTNLIST', 'MOBLIST', or 'STALIST' create ACTNRECl's, MOBRECl's, or STARECl's respectively. These are the top-level operational segment types of the INTERROGATOR. Currently, MOBRECl's only check the WEIGHT attribute of those mobile entities having that attribute to insure that it has weight units (e.g., pounds), and STARECl's perform no checks on the stationary entity records. However, if more detailed investigation of these records is required later, the mechanism is available.

2. ACTNRECl Rules

The ACTNRECl rules perform the basic checks on action records to insure that required attributes are present, and they also create other segments to conduct more detailed inspection of some of the attributes which are present.

As shown in the example problem, a single word--an action verb--is sufficient to start a problem description using the INTERROGATOR. It is also possible to use an action verb in such a way that its subject will not be clear to the decoding rules so that the AGENT or GOAL attribute is not set and, in turn, the MOBENATR attribute is not set. In either case, the first ACTNRECl rule detects the absence of the MOBENATR attribute in the original action record

(hence "¬ MOBENATR(CENTY)" instead of just "¬ MOBENATR"), and creates an INTSENT1 segment which is a copy of the ACTNREC1 with several other attributes added to meet requirements of the English encoding rules for producing a question of the form, WHAT ARRIVES?, and of the decoding rules for accepting the reply. Note, for instance, that the SUPSET attribute of MEMORY is set to 'MOBENTY' so that the user's reply can be tested immediately by the decoding rules to insure that it is in the set of mobile entities. Thus, an answer of "The station arrives" would result in a reply originating from the decoding rules of THAT IS NOT A REASONABLE REPLY. TRY AGAIN! The INTSENT1 segment type and the others which provide the interface with the English encoding and decoding rules will be discussed in considerably more detail after the other INTERROGATOR rules are discussed.

It was mentioned above that in the process of checking the SUCC attribute of an action record, any other action records encountered are checked only to determine if they have a MOBENATR attribute. This limited inspection is accomplished by setting the LIMITCHK indicator and depending on the second ACTNREC1 rule to stop the inspection. This limited checking is done to insure that the verb will have a subject as required by the English encoding rules. Furthermore, it is generally the case that such actions would not be specified in detail by the user when first mentioned but rather at some later time. Therefore, this rule helps the INTERROGATOR conduct the conversation in much the same way as the user would input the problem in purely narrative fashion.

The next ACTNREC1 rule determines if the LOCATION attribute is present in the original action record. Here, SUPSET(MEM) is set to 'LOCDESCR' to allow only responses that start with words such as "at" or "in," such as "at the station," to a question of the form, WHERE DO THE CARS ARRIVE? as produced by this rule.

The times associated with actions are of two types: an interevent time, or IETM, is associated with events, such as "arrive" and "enter," and a DURATION is used with activities, such as "service" and "unload." The next five rules focus on these times.

Rule 12 inspects the actions "arrive" and "enter" to determine if they have an IETM attribute and asks a question such as HOW OFTEN DO THE CARS ARRIVE AT THE STATION? if one is not present. The addition of new verbs in the same category ("approach," for instance) would pose no particular problems, for they could either be handled as individual cases as "arrive" and "enter" are now, or they could be grouped into a single set using the SUP principle discussed earlier. An example of this second approach can be seen below in the rule which tests for the presence of a DURATION attribute. Numerous other examples occur throughout the INTERROGATOR.

Note that this rule tests for the presence of the IETM attribute in the original record whereas the next rule says that if the ACTNREC1 itself has an IETM attribute, then two new segments are to be created. The purpose of the first segment, the QUANREC, is to conduct the checks discussed above to insure that the only units used are time units. As shown in the example problem, complex specification of an IETM is possible as long as whenever units are expressed, they are times. The second segment, the ACTNREC2, is a copy of the

ACTNREC1 but with the IETM attribute removed. This is a contingency segment: if the IETM attribute passes the tests posed for it, this ACTNREC2 will become an ACTNREC1 (see Rules 21 and 22) which will continue the testing process; otherwise, it does nothing. Note that if this segment does become an ACTNREC1, it will pass both of the IETM tests: IETM(CENTY) is true since Rule 12 guaranteed that the original IPR record has an IETM, and Rule 13 will not be satisfied because when the ACTNREC2 was created, its IETM attribute was removed.

The presence of the DURATION attribute is tested for in much the same way as IETM, except that actions to which this rule applies are tested by traversing their SUP chain to determine if they are in the set 'ACTIVITY'. The next two rules test the DURATION attribute if it is present. The first of these rules tests conditional durations to make sure that the object on which the condition is based is a stationary entity. This is a requirement of the current version of NLPQ. As shown in the example terminal session, a statement such as "the cars are serviced at the pump until a truck is available" is not allowed because a truck is not a stationary entity. The second rule tests the other DURATION attributes for the proper units using the same procedure as the IETM rule. Once again, an ACTNREC2 is created to continue testing if the consistency tests are passed.

The user may subclassify mobile entities by use of an assignment distribution, such as "40 percent of the vehicles are cars and 60 percent are trucks." In NLPQ, such an assignment distribution would be a record with a SUP of 'TYPDIST' pointed to by the ASNDISTR attribute of an action record. Since it is possible for a

user to specify an assignment distribution whose percentages do not add up to 100 or which has some other flaw, the INTERROGATOR checks for such conditions with the next rule. If the ASNDISTR attribute is present, a REC1 segment and an ACTNREC2 which is a copy of the ACTNREC1 with the ASNDISTR attribute removed are created. The ACTNREC2 segment serves the same purpose as those discussed previously. The REC1 segment is a copy of the ASNDISTR attribute with several of the attributes of the ACTNREC1. As will be seen below, REC1 segments are created from several different locations, but each has the same basic characteristic: each has a sequence of numbered attributes, starting with 101, whose values are pointers to some X and Y values grouped as pairs. In the case of assignment distributions, the X values, pointed to by the odd numbered attributes, are numbers which represent percentages, while the Y values are mobile entities such as cars and trucks. An example of a 'TYPDIST' record is shown in Figure 2. The REC1 rules check to make sure all the points are present, and then, in the case of a 'TYPDIST', create a REC3 which checks the numbers representing the cumulative percentages to insure that the last value is 100 percent.

All actions except "leave" require a SUCC attribute which indicates what happens after that action is complete. The presence of the SUCC attribute in the original record in the IPR is tested for in much the same way as the previous rules tested for other attributes. Since a great variety of possibilities exist for successors to an action, once the SUCC attribute is determined to be present, a SCSRREC1 segment which is a copy of the SUCC attribute is created to conduct detailed tests on the successor. Once again, an ACTNREC2 is

created to continue testing on the ACTNREC1 if the SUCC attribute is found to be satisfactory. The rules for SCSRREC1 will be discussed below.

The final ACTNREC1 rule is reached only when all the tests have been satisfied. This final rule creates an ICHECKED segment which in turn sets the CHECKED attribute on the original action record in the IPR. As mentioned above, this insures that when records further down the 'ACTNLIST' are being inspected, time is not wasted checking action records which have already been found to be satisfactory. However, if any changes are made to an action record after its CHECKED indicator has been set, the CHECKED indicator is turned off from the decoding rules so that the changes will be inspected by the INTERROGATOR.

3. ACTNREC2 and ACTNREC3 Rules

The next two rules act as filters for further testing of an action record. ACTNREC2's are created at the same time as QUANREC's, REC1's, and SCSRREC1's but are behind these segments on the stack. When an ACTNREC2 comes off the stack to be processed, it will create an ACTNREC1 which is a copy of the ACTNREC2 if the QUESTSW indicator of MEMORY is not set. This ACTNREC1 goes to the top of the ACTNREC1 rules to be processed, and, as has been shown, it will pass the point at which it was created, thus continuing the testing of the action record. If, however, QUESTSW(MEM) is set, no further testing is desired since some question has just been asked. All such pending segments go to NULL by causing execution of some rule such as Rule 22. Similar techniques are used below for other recursive situations.

The ACTNREC3 rules were discussed briefly above. Their purpose is to detect actions which have no predecessor. Since there is no attribute indicating the predecessor (primarily because several unrelated actions may be the predecessors of a single action), the problem is to detect actions which cannot be reached through the SUCC attribute of some other action. As was mentioned before, during the investigation of the SUCC attribute of each action on the 'ACTNLIST', all actions encountered have their REACHED indicator set to signify that problem flow to them can occur from some other action. ACTNREC3 segments are created only after all the actions on the 'ACTNLIST' have been inspected and found satisfactory by ACTNREC1's. Any action record whose REACHED indicator is not set, and whose SUP is not 'ARRIV' or 'ENTER' will cause the ACTNREC3 which is a copy of it to satisfy Rule 23 and ask a question of the form, BEFORE BEING SERVICED, WHAT DO THE CARS DO? This insures that flow can occur through all parts of the system.

4. MOBREC1, STAREC1, MEMREC1, and ICHECKED Rules

The only test currently being conducted on the mobile entities of the system is a check on the units of the weight attribute of a MOBREC1 segment if it is present. Rule 25 conducts this check in a manner analogous to Rules 13 and 16. The system can easily be expanded to investigate other attributes if the need arises. No checks are currently necessary for stationary entities, but once again the mechanism is available if needed.

The MEMREC1 segment is somewhat different from the other major segment types discussed so far because it is not a copy of any part of the IPR. It is merely a vehicle for testing the attributes

PROBTIME and TIMUNIT of MEMORY. Rule 28 insures that a MEMREC1 does no checking if a question has just been asked. Rule 29 requires that the period of simulated time is specified for the problem. Rule 30 asks for the basic time unit to be used in the model. Note that because of the order in which segments are created by Rule 1, this is the last condition checked by the INTERROGATOR.

As mentioned previously, some efficiency is gained by not rechecking action records which have already been found satisfactory. The same is true of mobile entity records and stationary entity records, so Rule 32 was written to consolidate the setting of the CHECKED attribute on records in the IPR. This rule, coupled with Rule 5, saves unnecessary inspection of records.

5. QUANREC Rules

The rules for QUANREC's conduct tests on the various quantities which appear in the system, whether expressed simply or in some complex fashion. It has already been stated that QUANREC segments are created by Rule 13 to investigate IETM, Rule 16 to investigate DURATION, and Rule 25 to investigate WEIGHT. It will be seen later that the mean, standard deviation, and half-range of various standard distributions are also inspected by QUANREC's, which are created by rules 39, 42, 44, and 68.

In each case, one of the attributes given to the segment is VALSET, or value set. Its value is set to 'ABSTIME' for those attributes concerned with time (e.g., DURATION) or 'ABSWEIT' for those concerned with weights. It is set at the top level and merely copied if recursion occurs, as in the case where, for example, the mean of an exponential distribution is normally distributed. This

insures that regardless of how many levels of recursion occur, when a value involving units is stated, those units are correct. Rules 33 through 37 will demonstrate this.

Rule 33 is the rule through which inspection of all quantities should ultimately terminate, for it is the one which verifies that the units are correct. This rule is satisfied if the SUP of the QUANREC is in the set indicated by the VALSET attribute of the QUANREC. Thus, if VALSET were set to 'ABSTIME' and the QUANREC being inspected were a copy of a record with a SUP of 'MINUTE', the rule would be satisfied since 'ABSTIME' is in the SUP chain of 'MINUTE'.

The next rule checks to see if the QUANREC is a copy of a record which defined one of the three standard distributions in the system. These are the exponential, normal, and uniform distributions. If the QUANREC is a copy of a record containing one of these three distributions, its SUP will be in the set 'STDIST' and this rule will result in a DSTRREC1 which will conduct further tests on the distribution record. At this time the CENTY attribute can be set to point to the record containing the distribution because we are now sure that the entire record will not have to be replaced, but rather that, in the worst case, only some attribute of that record will have to be provided or replaced. This principle is applied throughout the INTERROGATOR: CENTY is changed to point to a new record only when we are reasonably sure that the record itself is basically sound, with perhaps some of its attributes missing or in need of correction.

Rule 35 handles a special case. NLPQ currently does not allow interevent times to be expressed in the form, "10 minutes for cars, and 20 minutes for trucks."

The next rule insures that all other type-tables are inspected in much the same fashion as those with a SUP of 'TYPDIST', as discussed under Rule 17 for ASNDISTR. In a record with a SUP of 'TYPTABL', the X values are mobile entities while the Y values are quantities. An example of a 'TYPTABL' record is shown in Figure 2. As will be seen in the REC1 and REC2 rules, the Y values eventually become QUANREC's so that they may be checked for correctness of units.

If a QUANREC reaches the last QUANREC rule, it is because the units are wrong. The sample problem includes an example of the statement and question produced by this rule when the mean of the exponential distribution for the interarrival time was stated as 7 pounds.

6. DSTRREC1, DSTRREC2, and DSTRREC3 Rules

The set of rules for DSTRREC1, DSTRREC2, and DSTRREC3 segments conduct the tests on the three types of standard distributions. Since all three require a mean, Rule 38 tests for its presence, and Rule 39 creates a QUANREC to conduct the check on the units of the mean. The response given by Rule 38 depends on what mode the user has selected. If he is operating in the abbreviated question-answer mode, where it is clear from the context what the question refers to, no statement will be made and only a question of the type, WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION? will be asked. On the other hand, if the user is operating in any other mode, a preliminary statement is made which is of the form, THE TIME TO SERVICE THE CARS AT THE STATION IS EXPONENTIALLY DISTRIBUTED. The above question is then asked.

When Rule 39 creates the QUANREC, it also creates a DSTRREC2, which is a copy of the DSTRREC1. The DSTRREC2 will continue the inspection of the distribution, provided that the QUANREC does not result in a question. Rule 40 will destroy the DSTRREC2 in that event. Once the mean is found to be satisfactory, however, the DSTRREC2 segment will check a normal distribution to be sure it has a standard deviation (Rule 41), or a uniform distribution to be sure it has a range (Rule 43), and then a QUANREC will be created to conduct the tests for correct units on whatever standard deviations or ranges are found (Rules 42 and 44). These rules are comparable to the DSTRREC1 rules discussed above. Finally, Rule 45 is necessary, since an exponential distribution has neither standard deviation nor range.

The two rules for DSTRREC3 segments help determine the difference between the normal question-answer mode and the expanded question-answer mode. If the user has described a large part of a problem before invoking the INTERROGATOR, he may find it beneficial to have a preliminary description which indicates which area of the problem the INTERROGATOR question is referencing. For example, he may have specified two normal distributions but without standard deviations in either case. Without the description, THE TIME FOR THE CARS TO BE SERVICED AT THE PUMP IS NORMALLY DISTRIBUTED, WITH A MEAN OF 6 MINUTES, the question, WHAT IS THE STANDARD DEVIATION OF THE NORMAL DISTRIBUTION? would be ambiguous. Rule 46 will produce this description. On the other hand, if the user is in the usual question-answer mode where the context indicates which normal distribution

is referred to, Rule 47 will be invoked to reduce the duplication of information. A similar technique will be seen in describing successors to actions in Rules 58 and 59. The example given in Appendix B is in the expanded mode, while the example in Appendix C is in the usual mode.

7. SCSRREC1 and SCSRREC2 Rules

Because of the large number of possible ways to express what happens after an action is completed, twelve rules are required to test the SUCC attribute of an action. These are the rules for SCSRREC1 and SCSRREC2. Reference to Figure 2 will help in understanding the various structures which may be pointed to by the SUCC attribute.

Those successors which depend on some condition such as the length of a line or the availability of some facility or storage to determine which of two actions take place are records which have a SUP attribute of 'QTYP', 'FTYP', or 'STYP' respectively. Among other attributes, each of these records should have an OPENACT attribute and a CLOSACT attribute which indicate the alternative actions to be taken. Since two sentences are generally required to state such successors, it is quite easy for the user to omit one of the actions or make a mistake in expressing it. Rules 49 and 50 each create two new SCSRREC1's, one of which is a copy of the OPENACT attribute or CLOSACT attribute of the original SCSRREC1 and the other of which is a copy of the original SCSRREC1 with the appropriate OPENACT or CLOSACT attribute removed. This recursive feature, coupled with Rule 48, allows the inspection of several different types of successors with the same rules.

For example, suppose a 'QTYP' successor without a CLOSACT attribute and without a location for the line whose length is the determining factor is to be tested by these rules. Since it has an OPENACT attribute, Rule 49 would create two new SCSRREC1's as described above. The first would be a copy of the OPENACT attribute and hence would be a copy of some action. It would fail the test of all the SCSRREC1 rules until it reached Rule 55 or Rule 56. If the action is one which has not yet been completely specified and hence does not have its CHECKED indicator set, Rule 55 will insure that the action at least has an AGENT or GOAL to serve as the subject in a sentence by creating an ACTNREC1 with its LIMITCHK indicator set. As was stated earlier, Rule 10 will cause this ACTNREC1 to disappear if it passes Rule 9. Rule 55 also sets the REACHED indicator on the original action record in the IPR to indicate that flow to that action can occur. If the action has already been checked, then Rule 56 is executed to set the REACHED indicator.

Once the SCSRREC1 which is a copy of the OPENACT has been successfully processed, the second SCSRREC1 starts down the rules. Since it now has neither an OPENACT nor a CLOSACT, it will not satisfy Rules 49 or 50, but because there is no location for the line and hence no SUCARG attribute, Rule 51 would be satisfied. This results in a statement of the general form, IF THE LENGTH OF THE LINE IS LESS THAN 4, THE CAR WILL BE SERVICED, followed by the question, WHERE IS THE LINE? If the user is in the normal question-answer mode, only the question will be produced. The SCSRREC2 created by Rule 51 determines which of the courses to follow in Rules 58 and 59.

After the location of the line is known and a new inspection is begun by the INTERROGATOR, the same procedure described above will be repeated except that now the SCSRREC1 with a SUP of 'QTYP' would pass Rule 51. If the location of the line was properly stated as a stationary entity, Rule 52 will also be passed. However, since the original record was missing a CLOSACT attribute, Rule 53 will be executed. This occurs because during the processing of 'QTYP', 'FTYP', or 'STYP' records, the decoding rules create a temporary attribute called ACT2 which is removed when both the OPENACT and CLOSACT have been processed. Its presence in the record indicates one of the two attributes has not been specified, and its value indicates which one. Rule 53 produces a SCSRREC2 followed by an INTSENT2 which ultimately produces a question of the form, OTHERWISE, WHAT DO THE CARS DO? As before, the SCSRREC2 produces a description or goes to NULL, depending on the mode the user has selected.

This example for a 'QTYP' record covers a majority of the SCSRREC1 rules. Most of them also apply to 'FTYP' and 'STYP' records in a similar manner. 'PTYP' and 'FRACTNL' records are two other types which may be successors. 'PTYP' records define successors of the type, "cars leave and trucks unload," while 'FRACTNL' records result in the type, "20 percent of the vehicles leave, and 80 percent unload." Both types consist of X and Y pairs similar to 'TYPDIST' and 'TYPTABL' described previously, so Rule 54 creates a REC1 which is a copy of the SCSRREC1 to continue the testing.

Of course, the simplest type of successor is an action: "After arriving at the station, the cars are serviced." These successors are considered by Rules 55 and 56, as described above.

8. REC1, REC2, REC3, and REC4 Rules

All records having sequentially numbered attributes which start with @101 and which represent X and Y pairs of some sort are inspected by REC1 segments. As mentioned above, these are produced as copies of 'TYPDIST', 'TYPTABL', 'PTYP', or 'FRACTNL' records so that the X and Y values are of different types. Therefore, the REC1 rules perform two basic functions: first, they check all four types to verify that an X or Y value is not missing, and second, they separate the various types for further testing.

As usual, Rule 60 insures that if a question has been asked, no further testing will occur. This rule is necessary since Rules 60 through 66 form a loop in which each numbered attribute from @101 to the last one, as indicated by the value of the XYLAST attribute, is tested for by one pass through the loop. The XYC attribute is the loop parameter, and it is increased by one at the end of each pass by Rule 66. Rules 63 and 64 detect a missing attribute and produce an appropriate statement and question. In both cases, because of the difficulty of inserting the missing point in the IPR and identifying the missing information to the user, he is asked to restate the information in its entirety. Rule 63 produces four segments, the first three of which result in a statement and a description which involves the 'TYPTABL'. The fourth, an INTSENT3, ultimately results in a question about the attribute of the action record from which the investigation began, such as HOW LONG ARE THE CARS SERVICED

AT THE STATION? Thus, the user is asked to restate the conditions which originally resulted in the 'TYPTABL', but with the missing information supplied. Rule 64 produces the same two phrases followed by a REC4 segment which is a copy of the REC1. In a similar fashion, the two REC4 rules and the ASNERR and SUCCERR rules produce a statement of the incomplete description, followed by either the clause, PLEASE RESPECIFY,or a question about what follows the action currently being considered.

If all the points are present, Rules 61 and 62 determine what further tests are to be conducted. If the SUP of the REC1 is 'TYPTABL', Rule 61 produces a REC2 which is a copy of the REC1. Otherwise, Rule 62 produces a REC3 which is a copy of the REC1. Those REC1's with SUP's of 'TYPDIST', 'FRACTNL', and 'PTYP' take this second route.

When the REC2 is created, the CENTY attribute is set to point to the original IPR record with a SUP of 'TYPTABL', and the value of the XYC attribute is set to 102. This allows Rules 67 and 68 to form a loop similar to the REC1 loop except that only the even-numbered attributes are looked at. These are all supposed to be values, so a QUANREC is created which is a copy of that attribute so that no matter how complex the description of the value, the QUANREC rules will detect any values which have incorrect units.

The REC3 created by Rule 62 has the value of its XYC attribute set to the value of XYLAST-1, which is the number of the last X value. The first test in Rule 69 will eliminate the 'PTYP' records from further testing since their X values are mobile entities and hence cannot have a SUP of 'DECIMAL'. However, those records with a SUP of 'TYPDIST' or 'FRACTNL' have X values with a SUP

of 'DECIMAL', so the second part of Rule 69 must be satisfied for a question to be avoided. That test determines if the last cumulative percentage value is "sufficiently close" to one, where "sufficiently close" is defined here as within 10/1000. If these criteria are not met, Rule 70 produces the statement, THE FOLLOWING PERCENTAGES DO NOT TOTAL 100:, followed by a REC4 which produces one of the two statements previously discussed.

The two rules for REC4 segment types serve to separate those with a SUP of 'TYPDIST' from those with a SUP of 'FRACTNL'. Rule 71 creates an ASNERR segment from REC4 segments with a SUP attribute of 'TYPDIST'. Rule 72 produces a SUCCERR segment from those with a SUP of 'FRACTNL'. The next two rules show how these two new segment types are applied. Before they are discussed, however, some mention should be made of the connection of the INTERROGATOR rules with the English encoding rules.

9. Rules Relating to Encoding and Decoding

The remaining six rules all produce segments whose rules are found in the English encoding rules. With the addition of the PHRASE segment type, these six represent the basic tools with which the INTERROGATOR produces statements and questions in English text. The attributes which these segments have depend on the requirements of the English encoding rules, and also on the requirements of the decoding rules. While it is beyond the scope of this paper to discuss the English encoding or the decoding rules in detail, the effect of certain attributes of these six segment types on what is produced by the encoding or decoding rules will be discussed below.

An ASNERR segment produces an ASNDESC segment which is a copy of the ASNDISTR attribute of the original action record, followed by the PHRASE which produces the request to the user that he PLEASE RESPECIFY. The ASNDESC segment type in the English encoding rules will produce a syntactically correct English description of the assignment distribution, even though the percentages do not total 100. The PHRASE segment accomplishes two things besides producing the phrase. Since the LASTME attribute of MEMORY is used to resolve pronoun references to mobile entities, it must be set. In this case, it is set to the value of the STRUCENTY attribute of the ASNERR segment. The value of that attribute is the mobile entity record of the mobile entity whose composition is being specified by the assignment distribution. For example, if the user was discussing vehicles when he first specified an incomplete assignment distribution, his response to the request to PLEASE RESPECIFY could be, "40 percent of them are cars, and 60 percent are trucks," and "them" would be associated with "vehicles" through LASTME(MEM). The second thing accomplished in the PHRASE segment is to remove the ASNDISTR attribute from the action record in the IPR. This insures that the new assignment distribution will be inserted properly in the IPR.

A SUCCERR segment can be created from Rule 72 or from Rule 52. The SUCCERR segment type takes advantage of the SUCCDESC segment type of the English encoding rules, which was designed to describe some action as being the successor to another action. The English encoding rule required some rewriting since it originally would not work unless both the OPENACT and CLOSACT

attributes were present. Because of the test on the right of the rule, this SUCCDESC will have either the OPENACT or the CLOSACT, but not both. The test on the right consists of the test "OPENACT. NE. 0" followed by another instruction to be executed if the test is passed.

The second segment produced by the SUCCERR rule is an ACTNREC1 with the SUCC attribute removed both from the ACTNREC1 and the IPR record of which it is a copy. When this segment is processed, it will produce an appropriate question of the form, AFTER BEING SERVICED, WHAT DO THE CARS DO?

The remaining four rules all produce sentences of various forms by creating a SENT segment to be processed further by the English encoding rules. The two INTSENT1 rules produce the general purpose sentences: those that ask questions about missing attributes of action records, those that describe an action in some detail before an INTSENT3 is used to ask a question about a missing attribute of some record pointed to directly or indirectly from the action record, and those that say that some condition can not exist and provide a reason. Rule 75 handles the somewhat special case of the INTSENT1 created to say something about a mobile entity. The sentence produced is essentially the same as those produced from Rule 76, but certain attributes of MEMORY must be treated differently. For instance, LASTME(MEM) is set to point to the IPR record containing information about the mobile entity being considered. LASTSE(MEM), which ordinarily points to the last stationary entity mentioned, and LASTLD(MEM), which points to the last location descriptor mentioned, (e. g., "at" or "in") are both removed. Then two attributes required for the decoding rules to handle the user's response are set.

CENTY(MEM) is set to point to the IPR record to be changed; the CENTY attribute of all segments is used to point to the IPR record which will be changed if an error is found. ATTRIB(MEM) is given the name of the attribute which is to be changed or added. QUESTSW (MEM) is also set to prevent any further questions from being asked.

The second INTSENT1 rule handles the more general cases. It, too, creates a SENT which is a copy of the INTSENT1, but some attributes are removed if they are present, and the attributes of MEMORY are set somewhat differently. The attributes removed if they are present are PRED, SUCC, and CONDITN. Each of these, if present, causes additional information to be present in the sentence which is not required for this situation. For example, if PRED is present, it will produce a phrase such as, AFTER BEING SERVICED. Similarly, SUCC produces phrases of the form, BEFORE BEING SERVICED, and CONDITN starts the sentence with a phrase beginning with IF, or with the word OTHERWISE. As in the case of Rule 75, Rule 76 also sets a number of indicators and attributes of MEMORY. QUESTSW(MEM) is set for the same reason as previously discussed. LASTME(MEM) is set to the value of the attribute named in attribute MOBENATR--in other words, LASTME(MEM) gets the value of attribute AGENT or GOAL, which will be a pointer to a mobile entity. LASTSE(MEM) points to the last stationary entity mentioned, as contained in the LOCOBJ attribute of the record pointed to by the LOCATION attribute of the SENT. LASTLD(MEM) has as its value the value of the SUP attribute of the record pointed to by LOCATION. This will be some location descriptor such as "at" or "in." CENTY(MEM) and ATTRIB(MEM) are set as in Rule 75, but with

the added condition on the right that if the value of attribute ATTRIB is equal to the value of the attribute MOBENATR, then the ATTRIB attribute is to be removed. The reason this is necessary deserves further explanation, particularly since it is directly related to the reason the INTSENT1 produces questions in some cases and statements in others.

The mechanism of the English encoding rules recognizes certain values of attribute ATTRIB as indications of special requirements for the sentence. For instance, if the value of ATTRIB is 'DURATION', the sentence produced will start, THE TIME.... As the INTERROGATOR was developed, this feature was used, but it also suggested a method for determining when a question should be asked. The test is that if the segment has an ATTRIB attribute but not an attribute of the name which is the value of ATTRIB, then a question is asked about that attribute. For example, if the value of ATTRIB is 'DURATION', but there is no DURATION attribute, the sentence produced is one like, HOW LONG ARE THE CARS SERVICED AT THE PUMP? rather than one beginning, THE TIME FOR THE CARS TO BE SERVICED AT THE PUMP IS.... The HOW LONG is produced from the INTRGPH attribute of the named record for DURATION. The default condition to be executed if the named record has no INTRGPH attribute is to produce the word, WHAT. For example, WHAT IS THE MEAN OF THE NORMAL DISTRIBUTION?

This technique works well in all cases except that of a missing mobile entity. In that case, ATTRIB equals 'AGENT' or 'GOAL' and the record in the IPR does not have the corresponding AGENT or GOAL attribute. The desired question is one such as, WHAT ARRIVES?

or, WHAT IS SERVICED? but the rules would produce a question about AGENT or GOAL. Thus, the effect of the condition on the right in Rule 76 is to remove ATTRIB in that particular case after it has been used to set ATTRIB(MEM). The setting of the QUESMK indicator and the various attributes in Rule 9 assures the proper question in this case.

One other feature of the two INTSENT1 rules should be mentioned. The sentences which state that some condition can not exist and then state the reason result from the CAN and NEG indicators being set and a REASON attribute being created when the INTSENT1 is produced. Although these are not visible in Rules 74 and 76, since the SENT segments produced are copies of the INTSENT1's, these indicators and attributes will be carried along to produce the proper statements.

The basic purpose of the INTSENT2 is to produce a question about what a mobile entity does before, after, or other than some other action. Rules 18, 23, and 53 produce these questions. A sample question might be AFTER BEING SERVICED, WHAT DO THE CARS DO? Rule 77 creates a SENT which is a copy of the INTSENT2. It has an INTRGPH attribute with the value of "WHAT," its subject is the appropriate mobile entity, and it sets three attributes of MEMORY and the QUESTSW(MEM) indicator essentially the same way as Rule 76. CENTY(MEM) and ATTRIB(MEM), as well as some other attributes, are set when the INTSENT2 is created because of differences in the way the sentences and responses are handled. In particular, Rule 23 is different because when the question BEFORE... WHAT DO THE... DO? is asked, it is not known which record in the IPR will have to be

modified or if a new one will be created whose successor will be the action being considered. Hence, CENTY(MEM) can not be set. Instead, SUCC(MEM) is set to point to the action under consideration so that it may be placed as the SUCC of the action the user mentions in his reply.

The final segment type which produces a SENT is the INTSENT3. Its purpose is to ask a question in conjunction with a statement produced by an INTSENT1. The statements are of two types: those that say some condition can not exist and those that provide a partial description of a distribution. In the first case, the question asked will be about an erroneous attribute while in the second it will be about an as yet unspecified attribute. In both cases, the question results from the technique discussed above of there being an ATTRIB attribute but not an attribute with the name contained in ATTRIB. The "-@ATTRIB" on the right of Rule 78 assures this condition. SUPSET(MEM) is set to 'VALU' so that user responses which are not in that set will be rejected by the decoding rules with the statement, THAT IS NOT A REASONABLE REPLY. TRY AGAIN! Finally, ATTRIB(MEM) is set to the value of ATTRIB.

V. CONCLUSIONS AND RECOMMENDATIONS

The current version of NLPQ represents a significant advance in the area of natural language man-machine interaction. It allows the user to describe a simple queuing problem in English, converts the description to an internal representation, and can produce an English text description of the problem and a GPSS simulation program to solve the problem. The addition of the INTERROGATOR to NLPQ has enhanced these capabilities. The INTERROGATOR guarantees that the IPR is prepared to produce a GPSS program, by detecting missing or erroneous information and asking questions of the user. It has also made NLPQ easier to use by making it possible to enter an entire problem simply by answering questions, where both the questions and answers may be fairly complex English statements.

The information missing from an action record which can be detected includes such things as not having a mobile entity associated with it, no location being stated, no interevent time or duration being specified, and no action or set of actions being specified to follow this action. Missing parameters of standard distributions are also detected. Similarly, various special purpose records are checked to verify that they contain all required attributes. Finally, certain problem information is determined to be present in MEMORY.

Erroneous information is detected in several places. The QUANREC segments specifically determine that when an attribute concerned with time or weight is being specified, all units are times or weights, respectively. When percentages are stated, a REC3

segment will verify that they total 100. Certain restrictions of the current version of NLPQ are also met by testing. Finally, each action is checked to be sure that it can be reached from some other action unless it is one which specifies entry into the system.

The INTERROGATOR is capable of expanding with NLPQ. As new classes of problems are allowed and new attributes are created in the IPR, new tests can be added to the INTERROGATOR rules. Another important feature is that, for the most part, the INTERROGATOR is not tied to any particular natural language, because its rules were written at the semological level.

Not all forms of possible answers to INTERROGATOR questions are currently allowed by the decoding rules. One recommended area of further research would be to expand the decoding rules to accept a larger set of possible answers.

Another area for further investigation is to add the capability to the INTERROGATOR of determining that each mobile entity follows a complete path through the system. This check, coupled with the current check on the flow of actions through the system, would provide a global check on the completeness of the problem specification.

One other desirable feature which could be added to the INTERROGATOR would be the capability of detecting when an assignment distribution should have been specified. The basic situation should not pose a particular problem, since the presence of a type-table makes an assignment distribution mandatory. However, additional inspections would have to be made to verify that the mobile entities specified in the assignment distribution correspond to those in the type-table. This would be particularly important where more than one type-table is included.

APPENDIX A

INTERROGATOR QUESTIONS AND ALLOWABLE RESPONSES

The questions that follow are representative of the questions that the INTERROGATOR can ask. The numbers of the rule or rules which produce the question or a similar question appear in parenthesis on the following line. Various situations which result in the same basic question are grouped together. Representative forms of allowable user responses are shown under each group of questions. An ellipsis (. . .) is used where it is obvious what possibilities are allowed in place of the ellipsis.

WHAT ARRIVES?
(9)
Car.
The vehicles.
The vehicles arrive.
The vehicles arrive at the station.

WHERE DO THE VEHICLES ARRIVE?
(11)
At a pump in the station.
At the station.
They arrive at the station.
Vehicles arrive at the station.

HOW OFTEN DO THE VEHICLES ARRIVE AT THE STATION?
(12)

THE TIME BETWEEN ARRIVALS OF THE VEHICLES AT THE STATION CAN NOT BE 10 POUNDS, BECAUSE THE UNITS ARE WRONG. HOW OFTEN DO THE VEHICLES ARRIVE AT THE STATION?
(37)

THE TIME BETWEEN ARRIVALS OF THE VEHICLES AT THE STATION CAN NOT BE 10 MINUTES FOR CARS AND 20 MINUTES FOR TRUCKS, BECAUSE OF THE COMPLEX INTEREVENT TIME SPECIFICATION. HOW OFTEN DO THE VEHICLES ARRIVE AT THE STATION?

(35)
10 minutes.
Every 10 minutes.
They arrive every 10 minutes.
Vehicles arrive every 10 minutes.
Exponential.
Exponentially distributed.
Normally distributed, with a mean of 7 minutes, and a standard deviation of 2 minutes.
The time between arrivals is ....

THE TIME BETWEEN ARRIVALS OF THE VEHICLES AT THE STATION IS EXPONENTIALLY DISTRIBUTED. WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION?

(38, 41, 43)

THE TIME BETWEEN ARRIVALS OF THE VEHICLES AT THE STATION CAN NOT BE EXPONENTIALLY DISTRIBUTED, WITH A MEAN OF 10 POUNDS, BECAUSE THE UNITS ARE WRONG. WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION?

(37)

THE TIME FOR THE VEHICLES TO UNLOAD AT THE DOCK IS UNIFORMLY DISTRIBUTED, WITH A MEAN WHICH IS NORMALLY DISTRIBUTED. WHAT IS THE MEAN OF THE NORMAL DISTRIBUTION?

(39, 41, 43)
4 minutes.
Normally distributed.
The time between arrivals is exponentially distributed, with a mean of 4 minutes.

HOW LONG ARE THE VEHICLES SERVICED AT THE PUMP?

(14)

THE VEHICLES CAN NOT BE SERVICED AT THE PUMP UNTIL THE TRUCK IS AVAILABLE, BECAUSE THE CONDITIONAL DURATION ENTITY MUST BE STATIONARY. HOW LONG ARE THE VEHICLES SERVICED AT THE PUMP?

(15)

THERE IS SOMETHING MISSING IN THE FOLLOWING STATEMENT: THE TIME FOR THE VEHICLES TO UNLOAD AT THE DOCK IS 6 MINUTES FOR CARS AND FOR TRUCKS. HOW LONG DO THE VEHICLES UNLOAD AT THE DOCK?

(63)
10 minutes.
For 10 minutes.
Exponentially distributed ....
They are serviced for 10 minutes.
The vehicles are serviced for 10 minutes.
The time for servicing is normally distributed ....
Until the dock is available.
Until the dock is not busy.
The time to service vehicles is 10 minutes for cars and 20 minutes for trucks.

AFTER ARRIVING AT THE STATION, WHAT DO THE VEHICLES DO?

(18)

THE ENTITY IN THE FOLLOWING CONDITION MUST BE A STATIONARY ENTITY: AFTER ARRIVING AT THE STATION, IF THE TRUCK IS BUSY, THE VEHICLE WILL LEAVE THE STATION. AFTER ARRIVING AT THE STATION, WHAT DO THE VEHICLES DO?

(52 combined with 74)

THERE IS SOMETHING MISSING IN THE FOLLOWING STATEMENT: CARS UNLOAD AND SOMETHING LEAVES. AFTER ARRIVING AT THE STATION, WHAT DO THE VEHICLES DO?

(64 combined with 74)
Leave.
They leave.
The cars are serviced.
After arriving, they are serviced at the pump.
20 percent of them unload and 80 percent leave the station.
The cars unload and the trucks leave.
If the length of the line at the pump is not less than 4, they leave the station.
If the dock is available, they unload at the dock; otherwise, they leave the station.

AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE IS NOT LESS THAN FOUR, THE VEHICLE WILL LEAVE THE STATION. WHERE IS THE LINE?

(51)
At the pump.

AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE AT THE PUMP IS LESS THAN 7, THE CAR WILL BE SERVICED. OTHERWISE, WHAT DO THE CARS DO?

(53)
Leave.
They leave.
The cars leave the station.
Otherwise, they leave the station.

THE FOLLOWING PERCENTAGES DO NOT TOTAL 100. 20 PERCENT OF THE VEHICLES ARE CARS AND 60 PERCENT ARE TRUCKS. PLEASE RESPECIFY.

(70 combined with 73)
40 percent of the vehicles are cars and 60 percent are trucks.
40 percent of them are cars and 60 percent are trucks.

BEFORE BEING SERVICED AT THE PUMP, WHAT DO THE VEHICLES DO?

(23)
Unload.
They unload at the dock.
Before being serviced, the cars unload at the dock.

HOW LONG SHALL THE SIMULATION BE RUN?

(29)
3 hours.
For 3 hours.
The simulation shall be run for 3 hours.

WHAT IS THE BASIC TIME UNIT TO BE USED IN THE MODEL?

(30)
30 seconds.
The time unit is 30 seconds.

APPENDIX B

A SAMPLE INTERROGATOR TERMINAL SESSION

```
i cms
CMS..VERSION 01/21/71

login 191
R; T=0.02/0.23 12.06.05

login 192 t,p
** T (192) READ-ONLY **
R; T=0.01/0.04 12.06.15

load nlp prnams decode encode lpr bitstuff
R; T=1.70/3.85 12.06.55

start
EXECUTION BEGINS...
ENTER OPTIONAL DATA
 &p &end
TYPE NUMBER OF INITIAL INPUT FILE

9

TYPE NUMBER OF NEXT INPUT FILE

2

     14011
TYPE NUMBER OF NEXT INPUT FILE

3

     15798
```

TYPE NUMBER OF NEXT INPUT FILE

t

decode:

Arrive.

Check the IPR, please.

WHAT ARRIVES?

Cars arrive.

WHERE DO THE CARS ARRIVE?

At the station.

HOW OFTEN DO THE CARS ARRIVE AT THE STATION?

The time between arrivals is exponentially distributed.

THE TIME BETWEEN ARRIVALS OF THE CARS AT THE STATION IS EXPONENTIALLY DISTRIBUTED. WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION?

7 pounds.

THE TIME BETWEEN ARRIVALS OF THE CARS AT THE STATION CAN NOT BE EXPONENTIALLY DISTRIBUTED, WITH A MEAN OF 7 POUNDS, BECAUSE THE UNITS ARE WRONG. WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION?

7 minutes.

AFTER ARRIVING AT THE STATION, WHAT DO THE CARS DO?

If the length of the line is less than 4, the cars are serviced.

AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE IS LESS THAN 4, THE CAR WILL BE SERVICED. WHERE IS THE LINE?

At a pump.

AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE AT THE PUMP IS LESS THAN 4, THE CAR WILL BE SERVICED. OTHERWISE, WHAT DO THE CARS DO?

They unload at the dock.

WHERE ARE THE CARS SERVICED?

At the pump.

HOW LONG ARE THE CARS SERVICED AT THE PUMP?

Service times are exponential.

THE TIME FOR THE CARS TO BE SERVICED AT THE PUMP IS EXPONENTIALLY DISTRIBUTED. WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION?

Normally distributed, with a mean of 3 minutes.

THE TIME FOR THE CARS TO BE SERVICED AT THE PUMP IS EXPONENTIALLY DISTRIBUTED, WITH A MEAN WHICH IS NORMALLY DISTRIBUTED, WITH A MEAN OF 3 MINUTES. WHAT IS THE STANDARD DEVIATION OF THE NORMAL DISTRIBUTION?

2 minutes.

AFTER BEING SERVICED AT THE PUMP, WHAT DO THE CARS DO?

If the dock is not available, they leave the station.

AFTER BEING SERVICED AT THE PUMP, IF THE DOCK IS BUSY, THE CAR WILL LEAVE THE STATION. OTHERWISE, WHAT DO THE CARS DO?

Otherwise, they unload at the dock.

HOW LONG DO THE CARS UNLOAD AT THE DOCK?

Until a truck is available.

THE CARS CAN NOT UNLOAD AT THE DOCK UNTIL THE TRUCKS ARE AVAILABLE, BECAUSE THE CONDITIONAL DURATION ENTITY MUST BE STATIONARY. HOW LONG DO THE CARS UNLOAD AT THE DOCK?

Until a window is available.

AFTER UNLOADING AT THE DOCK, WHAT DO THE CARS DO?

They load at the window.

HOW LONG DO THE CARS LOAD AT THE WINDOW?

For 3 minutes.

AFTER LOADING AT THE WINDOW, WHAT DO THE CARS DO?

20 percent of the cars leave the station, and 60 percent unload

at the depot.

THE FOLLOWING PERCENTAGES DO NOT TOTAL 100: AFTER LOADING AT THE WINDOW, 20 PERCENT OF THE CARS LEAVE THE STATION, AND 60 PERCENT UNLOAD AT THE DEPOT. AFTER LOADING AT THE WINDOW, WHAT DO THE CARS DO?

40 percent of the cars leave the station, and 60 percent unload

at the depot.

HOW LONG DO THE CARS UNLOAD AT THE DEPOT?

6 minutes.

AFTER UNLOADING AT THE DEPOT, WHAT DO THE CARS DO?

They leave the station.

HOW LONG SHALL THE SIMULATION BE RUN?

For 100 minutes.

WHAT IS THE BASIC TIME UNIT TO BE USED IN THE MODEL?

30 seconds.

THE PROBLEM STATEMENT IS COMPLETE.

State the problem in English.

THE CARS ARRIVE AT THE STATION. THE TIME BETWEEN ARRIVALS OF THE CARS AT THE STATION IS EXPONENTIALLY DISTRIBUTED, WITH A MEAN OF 7 MINUTES. AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE AT THE PUMP IS LESS THAN 4, THE CAR WILL BE SERVICED AT THE PUMP. OTHERWISE, THE CAR WILL UNLOAD AT THE DOCK UNTIL THE WINDOW IS AVAILABLE. THE TIME FOR THE CARS TO BE SERVICED AT THE PUMP IS EXPONENTIALLY DISTRIBUTED, WITH A MEAN WHICH IS NORMALLY DISTRIBUTED, WITH A MEAN OF 3 MINUTES AND A STANDARD DEVIATION OF 2 MINUTES. AFTER BEING SERVICED AT THE PUMP, IF THE DOCK IS BUSY, THE CAR WILL LEAVE THE STATION. OTHERWISE, THE CAR WILL UNLOAD AT THE DOCK UNTIL THE WINDOW IS AVAILABLE. AFTER UNLOADING AT THE DOCK, THE CARS LOAD AT THE WINDOW FOR 3 MINUTES. AFTER LOADING AT THE WINDOW, 40 PERCENT OF THE CARS LEAVE THE STATION, AND THE REST UNLOAD AT THE DEPOT FOR 6 MINUTES. AFTER UNLOADING AT THE DEPOT, THE CARS LEAVE THE STATION.

THE SIMULATION IS TO BE RUN FOR 100 MINUTES, USING A BASIC TIME UNIT OF 30 SECONDS.

Write a GPSS program.

```
        SIMULATE
        RMULT       277,423,715,121,655,531,999,813
 STAT1  EQU         1,F,Q
 PUMP2  EQU         2,F,Q
 DOCK3  EQU         3,F,Q
 WIND4  EQU         4,F,Q
 DEPO5  EQU         5,F,Q
 CAR1   EQU         1,T
 1      TABLE       M1,,1,2
 1      FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.39/
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9997,8/
 2      FUNCTION    RN2,C29
0,-3/.012,-2.25/.027,-1.93/.043,-1.72/.062,-1.54/.084,-1.38/
.104,-1.26/.131,-1.12/.159,-1/.187,-.89/.23,-.74/.267,-.62/.334,-.43/
.432,-.17/.5,0/.568,.17/.666,.43/.732,.62/.77,.74/.813,.89/.841,1/
.869,1.12/.896,1.26/.916,1.38/.938,1.54/.957,1.72/.973,1.93/
.988,2.25/1,3/
 3      FUNCTION    RN3,D2
.400,ACT4/1.000,ACT6/
 1      FVARIABLE   6+4*FN2
*
*       THE CARS ARRIVE AT THE STATION.
        GENERATE    14,FN1
        ASSIGN      1,1
        TEST L      Q$PUMP2,4,ACT3
```

```
*
*       THE CARS ARE SERVICED AT THE PUMP.
 ACT2   QUEUE       PUMP2
        SEIZE       PUMP2
        DEPART      PUMP2
        ADVANCE     V1,FN1
        RELEASE     PUMP2
        GATE NU     DOCK3,ACT4
*
*       THE CARS UNLOAD AT THE DOCK.
 ACT3   QUEUE       DOCK3
        SEIZE       DOCK3
        DEPART      DOCK3
        GATE NU     WIND4
        RELEASE     DOCK3
        TRANSFER    ,ACT5
*
*       THE CARS LEAVE THE STATION.
 ACT4   TABULATE    P1
        TERMINATE
*
*       THE CARS LOAD AT THE WINDOW.
 ACT5   QUEUE       WIND4
        SEIZE       WIND4
        DEPART      WIND4
        ADVANCE     6
        RELEASE     WIND4
        TRANSFER    ,FN3
```

```
*
*       THE CARS UNLOAD AT THE DEPOT.
 ACT6   QUEUE       DEPO5
        SEIZE       DEPO5
        DEPART      DEPO5
        ADVANCE     12
        RELEASE     DEPO5
        TRANSFER    ,ACT4
*
*       TIMING LOOP
        GENERATE    200
        TERMINATE   1
        START       1
        END
```

APPENDIX C

SAMPLE SESSION FOR THE PROBLEM OF FIGURE 2

```
login 191
R; T=0.02/0.14 12.23.08

login 192 t,p
** T (192) READ-ONLY **
R; T=0.01/0.10 12.23.21

load nlp prnams decode encode lpl@r bitstuff
R; T=1.70/3.93 12.24.29

start
EXECUTION BEGINS...
ENTER OPTIONAL DATA
 &p &end
TYPE NUMBER OF INITIAL INPUT FILE

9

TYPE NUMBER OF NEXT INPUT FILE

3

     15812
TYPE NUMBER OF NEXT INPUT FILE

t
```

decode:

Vehicles arrive.

60 percent of the vehicles are cars and one fourth

of them are trucks.

Ask questions, please.

WHERE DO THE VEHICLES ARRIVE?

at the station.

HOW OFTEN DO THE VEHICLES ARRIVE AT THE STATION?

Normally distributed.

WHAT IS THE MEAN OF THE NORMAL DISTRIBUTION?

8 minutes.

WHAT IS THE STANDARD DEVIATION OF THE NORMAL DISTRIBUTION?

2 pounds.

THE TIME BETWEEN ARRIVALS OF THE VEHICLES AT THE STATION CAN NOT BE NORMALLY DISTRIBUTED, WITH A MEAN OF 8 MINUTES AND A STANDARD DEVIATION OF 2 POUNDS, BECAUSE THE UNITS ARE WRONG. WHAT IS THE STANDARD DEVIATION OF THE NORMAL DISTRIBUTION?

2 minutes.

THE FOLLOWING PERCENTAGES DO NOT TOTAL 100: 60 PERCENT OF THE VEHICLES ARE CARS, AND 25 PERCENT ARE TRUCKS. PLEASE RESPECIFY.

Three quarters of the vehicles are cars and one fourth of them are trucks.

AFTER ARRIVING AT THE STATION, WHAT DO THE VEHICLES DO?

If the length of the line is not less than two, they leave the station.

WHERE IS THE LINE?

At the pump in the station.

OTHERWISE, WHAT DO THE VEHICLES DO?

They are serviced at the pump.

HOW LONG ARE THE VEHICLES SERVICED AT THE PUMP IN THE STATION?

Service times are exponential.

WHAT IS THE MEAN OF THE EXPONENTIAL DISTRIBUTION?

5 minutes for cars and 9 minutes for trucks.

AFTER BEING SERVICED AT THE PUMP IN THE STATION, WHAT DO THE VEHICLES DO?

They leave the station.

HOW LONG SHALL THE SIMULATION BE RUN?

For 8 hours.

WHAT IS THE BASIC TIME UNIT TO BE USED IN THE MODEL?

30 seconds.

THE PROBLEM STATEMENT IS COMPLETE.

Describe the problem in English.

THE VEHICLES ARRIVE AT THE STATION. THE TIME BETWEEN ARRIVALS OF THE VEHICLES AT THE STATION IS NORMALLY DISTRIBUTED, WITH A MEAN OF 8 MINUTES AND A STANDARD DEVIATION OF 2 MINUTES. 75 PERCENT OF THE VEHICLES ARE CARS, AND THE REST ARE TRUCKS. AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE AT THE PUMP IN THE STATION IS LESS THAN 2, THE VEHICLE WILL BE SERVICED AT THE PUMP IN THE STATION. OTHERWISE, THE VEHICLE WILL LEAVE THE STATION. THE TIME FOR THE VEHICLES TO BE SERVICED AT THE PUMP IN THE STATION IS EXPONENTIALLY DISTRIBUTED, WITH A MEAN OF 5 MINUTES FOR THE CARS, AND 9 MINUTES FOR THE TRUCKS. AFTER BEING SERVICED AT THE PUMP IN THE STATION, THE VEHICLES LEAVE THE STATION.

THE SIMULATION IS TO BE RUN FOR 8 HOURS, USING A BASIC TIME UNIT OF 30 SECONDS.

Write a GPSS program for this problem.

```
        SIMULATE
        RMULT       277,423,715,121,655,531,999,813
 STAT1  EQU         1,F,Q
 PUMP2  EQU         2,F,Q
 CAR2   EQU         2,T
 2      TABLE       M1,1,1,2
 TRUC3  EQU         3,T
 3      TABLE       M1,1,1,2
 1      FUNCTION    RN1,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.39/
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2/
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9997,8/
 2      FUNCTION    RN2,C29
0,-3/.012,-2.25/.027,-1.93/.043,-1.72/.062,-1.54/.084,-1.38/
.104,-1.26/.131,-1.12/.159,-1/.187,-.89/.23,-.74/.267,-.62/.334,-.43/
.432,-.17/.5,0/.568,.17/.666,.43/.732,.62/.77,.74/.813,.89/.841,1/
.869,1.12/.896,1.26/.916,1.38/.938,1.54/.957,1.72/.973,1.93/
.988,2.25/1,3/
 3      FUNCTION    P1,D2
CAR2,10/TRUC3,18/
 4      FUNCTION    RN3,D2
.750,CAR2/1.000,TRUC3/
 1      FVARIABLE   16+4*FN2
```

```
*
*       THE VEHICLES ARRIVE AT THE STATION.
        GENERATE    V1
        ASSIGN      1,FN4
        TEST L      Q$PUMP2,2,ACT2
        TRANSFER    ,ACT3
*
*       THE VEHICLES LEAVE THE STATION.
 ACT2   TABULATE    P1
        TERMINATE
*
*       THE VEHICLES ARE SERVICED AT THE PUMP IN THE STATION.
 ACT3   QUEUE       PUMP2
        SEIZE       PUMP2
        DEPART      PUMP2
        ADVANCE     FN3,FN1
        RELEASE     PUMP2
        TRANSFER    ,ACT2
*
*       TIMING LOOP
        GENERATE    960
        TERMINATE   1
        START       1
        END
```

APPENDIX D

THE INTERROGATOR RULES

NAMED RECORDS:

```
REASON1 (CHARS=" THE CONDITIONAL DURATION ENTITY MUST BE STATIONARY")
REASON2  (CHARS=" THE UNITS ARE WRONG")
REASON3  (CHARS=" OF THE COMPLEX INTEREVENT TIME SPECIFICATION")
```

SEMOLOGY FOR ENCODING:

```
(1)   INTERROGATOR   -->
                SETAGL(%ACPTR(MEM),LC=11)   ...
                RECLISTI(%'ACTNLIST',LIST='ACTNLIST',LC=11)   ...
                RECLISTI(%'ACTNLIST',LIST='ACTNLIST',LC=11,K=3)   ...
                RECLISTI(%'MOBLIST',LIST='MOBLIST',LC=11)   ...
                RECLISTI(%'STALIST',LIST='STALIST',LC=11)   ...
                MEMREC1

(2)   RECLISTI(LC.LE.LASTREC,¬QUESTSW(MEM))   -->
                RECI(%∂LC(RECLISTI)(RECLISTI),K(RECLISTI),LIST(RECLISTI),
                     CENTY=∂LC(RECLISTI)(RECLISTI))   ...
                RECLISTI(LC=LC+1)

(3)   RECLISTI   -->   NULL

(4)   RECI(LIST.EQ.'ACTNLIST',K.EQ.3)   -->   ACTNREC3(%RECI)

(5)   RECI(CHECKED)   -->   NULL

(6)   RECI(LIST.EQ.'ACTNLIST')   -->   ACTNREC1(%RECI)

(7)   RECI(LIST.EQ.'MOBLIST')   -->   MOBREC1(%RECI)

(8)   RECI(LIST.EQ.'STALIST')   -->   STAREC1(%RECI)
```

```
(9)     ACTNREC1(¬MOBENATR(CENTY))   -->
                INTSENT1(%ACTNREC1,SUBJECT='WHATREC',SING,
                         MOBENATR=AGORGL($(SEG)),ATTRIB=MOBENATR,
                         SUPSET(MEM)='MOBENTY',QUESMK,
                         ¬MOBENATR,MOBENATR='AGENT',ATTRIB='AGENT')

(10)    ACTNREC1(LIMITCHK)   -->   NULL

(11)    ACTNREC1(¬LOCATION(CENTY))   -->
                INTSENT1(%ACTNREC1,ATTRIB='LOCATION',
                         SUPSET(MEM)='LOCDESCR')

(12)    ACTNREC1(¬IETM(CENTY),'ARRIV'|'ENTER')   -->
                INTSENT1(%ACTNREC1,ATTRIB='IETM')

(13)    ACTNREC1(IETM)   -->
                QUANREC(%IETM(ACTNREC1),CENTY(ACTNREC1),AENTY=CENTY,
                        ATTRIB='IETM',VALSET='ABSTIME',ATTRIB1='IETM')   ...
                ACTNREC2(%ACTNREC1,-IETM)

(14)    ACTNREC1(¬DURATION(CENTY),$'ACTIVITY')   -->
                INTSENT1(%ACTNREC1,ATTRIB='DURATION')

(15)    ACTNREC1(DURATION$'COND',CONDENTY(DURATION)¬$'STATENTY')   -->
                INTSENT1(%CENTY(ACTNREC1),CENTY(ACTNREC1),ATTRIB='DURATION',
                         CAN,NEG,REASON=CHARS('REASON1'))   ...
                INTSENT3(%CENTY(ACTNREC1),ATTRIB='DURATION')

(16)    ACTNREC1(DURATION,DURATION¬$'COND')   -->
                QUANREC(%DURATION(ACTNREC1),CENTY(ACTNREC1),AENTY=CENTY,
                        ATTRIB='DURATION',VALSET='ABSTIME',
                        ATTRIB1='DURATION')   ...
                ACTNREC2(%ACTNREC1,-DURATION)

(17)    ACTNREC1(ASNDISTR)   -->
                REC1(%ASNDISTR(ACTNREC1),CENTY(ACTNREC1),AENTY=CENTY,
                     XYC=101)   ...
                ACTNREC2(%ACTNREC1,-ASNDISTR)

(18)    ACTNREC1(¬SUCC(CENTY),¬'LEAV')   -->
                INTSENT2('DO',PRED=ACTNREC1,-IETM(PRED),-DURATION(PRED),
                         ATTRIB(MEM)='SUCC',PRED(MEM)=CENTY(ACTNREC1),
                         CENTY(MEM)=CENTY(ACTNREC1),AENTY=CENTY(ACTNREC1))

(19)    ACTNREC1(SUCC)   -->
                SCSRREC1(%SUCC(ACTNREC1),CENTY(ACTNREC1),AENTY=CENTY,
                         ENTY=SUCC(ACTNREC1),ATTRIB1='SUCC')   ...
                ACTNREC2(%ACTNREC1,-SUCC)
```

```
(20)    ACTNREC1    -->    ICHECKED(%ACTNREC1)

(21)    ACTNREC2(¬QUESTSW(MEM))    -->    ACTNREC1(%ACTNREC2)

(22)    ACTNREC2    -->    NULL

(23)    ACTNREC3(¬REACHED,¬'ARRIV',¬'ENTER')    -->
                INTSENT2('DO',SUCC=ACTNREC3,-IETM(SUCC),-DURATION(SUCC),
                          ATTRIB(MEM)='PRED',SUCC(MEM)=CENTY(ACTNREC3),
                          AENTY=CENTY(ACTNREC3))

(24)    ACTNREC3    -->    NULL

(25)    MOBREC1(WEIGHT(CENTY))    -->
                QUANREC(%WEIGHT(MOBREC1),CENTY(MOBREC1),AENTY=CENTY,
                          ATTRIB='WEIGHT',VALSET='ABSWEIT',ATTRIB1='WEIGHT')

(26)    MOBREC1    -->    ICHECKED(%MOBREC1)

(27)    STAREC1    -->    ICHECKED(%STAREC1)

(28)    MEMREC1(QUESTSW(MEM))    -->    NULL

(29)    MEMREC1(¬PROBTIME(MEM))    -->
                PHRASE(CHARS=" HOW LONG SHALL THE SIMULATION BE RUN?",
                          ATTRIB(MEM)='PROBTIME',CENTY(MEM)=MEM,
                          SUPSET(MEM)='ABSTIME',QUESTSW(MEM))

(30)    MEMREC1(¬TIMUNIT(MEM))    -->
                PHRASE(CHARS=" WHAT IS THE BASIC TIME UNIT TO BE USED")  ...
                PHRASE(CHARS=" IN THE MODEL",
                          ATTRIB(MEM)='TIMUNIT',CENTY(MEM)=MEM,
                          SUPSET(MEM)='ABSTIME',QUESTSW(MEM))

(31)    MEMREC1    -->    NULL

(32)    ICHECKED    -->    NULL(CHECKED(CENTY(ICHECKED)))

(33)    QUANREC($VALSET)    -->    NULL

(34)    QUANREC($'STDIST')    -->
                DSTRREC1(%QUANREC,CENTY=@ATTRIB1(SEG)(CENTY))

(35)    QUANREC('TYPTABL',ATTRIB.EQ.'IETM')    -->
                INTSENT1(%AENTY(QUANREC),AENTY(QUANREC),CENTY(QUANREC),
                          ATTRIB(QUANREC),CAN.NEG,REASON=CHARS('REASON3')) ...
                INTSENT3(%CENTY(QUANREC),ATTRIB=ATTRIB1(QUANREC))
```

```
(36)    QUANREC('TYPTABL')   -->   REC1(%QUANREC,XYC=101)
(37)    QUANREC   -->
                INTSENT1(%AENTY(QUANREC),AENTY(QUANREC),ATTRIB(QUANREC),
                         CENTY(QUANREC),CAN,NEG,REASON=CHARS('REASON2')) ...
                INTSENT3(%CENTY(QUANREC),ATTRIB=ATTRIB1(QUANREC))

(38)    DSTRREC1(¬MEAN)   -->
                DSTRREC3(%DSTRREC1)   ...
                INTSENT3(%DSTRREC1,ATTRIB='MEAN')

(39)    DSTRREC1(MEAN)   -->
                QUANREC(%MEAN(DSTRREC1),AENTY(DSTRREC1),ATTRIB(DSTRREC1),
                         VALSET(DSTRREC1),ATTRIB1='MEAN',
                         CENTY(DSTRREC1))   ...
                DSTRREC2(%DSTRREC1)

(40)    DSTRREC2(QUESTSW(MEM))   -->   NULL

(41)    DSTRREC2('NORMAL',¬STDEV)   -->
                DSTRREC3(%DSTRREC2)   ...
                INTSENT3(%DSTRREC2,ATTRIB='STDEV')

(42)    DSTRREC2(STDEV)   -->
                QUANREC(%STDEV(DSTRREC2),AENTY(DSTRREC2),ATTRIB(DSTRREC2),
                        VALSET(DSTRREC2),ATTRIB1='STDEV',
                        CENTY(DSTRREC2))

(43)    DSTRREC2('UNIFORM',¬RANGE)   -->
                DSTRREC3(%DSTRREC2)   ...
                INTSENT3(%DSTRREC2,ATTRIB='RANGE')

(44)    DSTRREC2(RANGE)   -->
                QUANREC(%RANGE(DSTRREC2),AENTY(DSTRREC2),ATTRIB(DSTRREC2),
                        VALSET(DSTRREC2),ATTRIB1='RANGE',
                        CENTY(DSTRREC2))

(45)    DSTRREC2   -->   NULL

(46)    DSTRREC3(¬QAMODE(MEM))   -->
                INTSENT1(%AENTY(DSTRREC3),ATTRIB(DSTRREC3),CENTY(DSTRREC3))

(47)    DSTRREC3   -->   NULL(QUESTSW(MEM),ATTRIB(MEM)=ATTRIB(DSTRREC3),
                              CENTY(MEM)=CENTY(DSTRREC3))

(48)    SCSRREC1(QUESTSW(MEM))   -->   NULL
```

```
(49)    SCSRREC1(OPENACT)   -->
            SCSRREC1(%OPENACT(SCSRREC1),AENTY(SCSRREC1),
                     CENTY=@ATTRIB1(SCSRREC1)(CENTY(SCSRREC1)),
                     ATTRIB1='OPENACT')   ...
            SCSRREC1(%SCSRREC1,-OPENACT)

(50)    SCSRREC1(CLOSACT)   -->
            SCSRREC1(%CLOSACT(SCSRREC1),AENTY(SCSRREC1),
                     CENTY=@ATTRIB1(SCSRREC1)(CENTY(SCSRREC1)),
                     ATTRIB1='CLOSACT')   ...
            SCSRREC1(%SCSRREC1,-CLOSACT)

(51)    SCSRREC1('QTYP',¬SUCARG)   -->
            SCSRREC2(%SCSRREC1)   ...
            SENT('BE',QUESTSW(MEM),INTRGPH=" WHERE",SUBJECT='LINEREC',
                 SUPSET(MEM)='LOCDESCR',ATTRIB(MEM)='SUCARG',
                 CENTY(MEM)=@ATTRIB1(SCSRREC1)(CENTY(SCSRREC1)))

(52)    SCSRREC1($'SUCDSCR2',SUCARG¬$'STATENTY')   -->
            PHRASE(CHARS=" THE ENTITY IN THE FOLLOWING CONDITION")   ...
            PHRASE(CHARS=" MUST BE A STATIONARY ENTITY:  ")   ...
            SUCCERR(%SCSRREC1)

(53)    SCSRREC1($'SUCDSCR2',ACT2)   -->
            SCSRREC2(%SCSRREC1)   ...
            INTSENT2('DO',AENTY(SCSRREC1),CONDITN='OTHERWIS',
                     ATTRIB(MEM)=ACT2(SCSRREC1),
                     CENTY(MEM)=@ATTRIB1(SCSRREC1)(CENTY(SCSRREC1)))

(54)    SCSRREC1(XYLAST)   -->   REC1(%SCSRREC1,XYC=101)

(55)    SCSRREC1($'ACTION',¬CHECKED)   -->
            ACTNREC1(%SCSRREC1,CENTY=@ATTRIB1(SEG)(CENTY(SEG)),
                     AENTY=CENTY,REACHED(AENTY),LIMITCHK)

(56)    SCSRREC1($'ACTION')   -->
            NULL(REACHED(@ATTRIB1(SCSRREC1)(CENTY(SCSRREC1))))

(57)    SCSRREC1   -->   NULL

(58)    SCSRREC2(¬QAMODE(MEM))   -->
            SUCCDESC(%@ATTRIB1(SCSRREC2)(CENTY(SCSRREC2)),
                     PRED=%AENTY(SCSRREC2),-IETM(PRED),-DURATION(PRED),
                     OPENACT.NE.0,-CLOSACT)

(59)    SCSRREC2   -->   NULL
```

```
(60)    REC1(QUESTSW(MEM))   -->   NULL
(61)    REC1(XYC.GT.XYLAST,'TYPTABL')   -->
                REC2(%REC1,CENTY=@ATTRIB1(SEG)(CENTY),XYC=102)

(62)    REC1(XYC.GT.XYLAST)   -->   REC3(%REC1,XYC=XYLAST-1)

(63)    REC1(¬@XYC,'TYPTABL')   -->
                PHRASE(CHARS=" THERE IS SOMETHING MISSING IN THE")   ...
                PHRASE(CHARS=" FOLLOWING STATEMENT:  ")   ...
                INTSENT1(%AENTY(REC1),AENTY(REC1),ATTRIB(REC1),
                        CENTY(REC1))   ...
                INTSENT3(%AENTY(REC1),ATTRIB(REC1))

(64)    REC1(¬@XYC)   -->
                PHRASE(CHARS=" THERE IS SOMETHING MISSING IN THE")   ...
                PHRASE(CHARS=" FOLLOWING STATEMENT:  ")   ...
                REC4(%REC1)

(65)    REC1(@XYC$'ACTION',¬CHECKED(@XYC))   -->
                ACTNREC1(%@XYC(REC1)(REC1),CENTY=@XYC(REC1)(REC1),
                        LIMITCHK)   ...
                REC1(XYC=XYC+1)

(66)    REC1   -->   REC1(XYC=XYC+1,@XYC$'ACTION',REACHED(@XYC))

(67)    REC2(XYC.GT.XYLAST|QUESTSW(MEM))   -->   NULL

(68)    REC2   -->
                QUANREC(%@XYC(REC2)(REC2),AENTY(REC2),ATTRIB(REC2),
                        VALSET(REC2),ATTRIB1=XYC(REC2),CENTY(REC2))   ...
                REC2(XYC=XYC+2)

(69)    REC3(SUP(@XYC).NE.'DECIMAL'|NUM(@XYC).GT.990,NUM(@XYC).LT.1010)   -->
                NULL
(70)    REC3   -->
                PHRASE(CHARS=" THE FOLLOWING PERCENTAGES DO NOT TOTAL 100:",
                        NOREST(MEM))   #   #   ...
                REC4(%REC3)

(71)    REC4('TYPDIST')   -->   ASNERR(%REC4)

(72)    REC4   -->   SUCCERR(%REC4)
```

```
(73)    ASNERR    -->   ASNDESC(%ASNDISTR(AENTY(ASNERR)))    ...
                  PHRASE(CHARS=" PLEASE RESPECIFY.",QUESTSW(MEM),
                         LASTME(MEM)=STRUCENTY(ASNERR),
                         -ASNDISTR(CENTY(ASNERR)))

(74)    SUCCERR    -->
                  SUCCDESC(%SUCC(AENTY(SUCCERR)),PRED=%AENTY(SUCCERR),
                         -IETM(PRED),-DURATION(PRED),OPENACT.NE.0,
                         -CLOSACT)    ...
                  ACTNREC1(%AENTY(SUCCERR),CENTY=AENTY(SUCCERR),-SUCC,
                         -SUCC(CENTY))

(75)    INTSENT1($'MOBENTY')    -->
                  SENT(%INTSENT1,QUESTSW(MEM),LASTME(MEM)=AENTY,-LASTSE(MEM),
                       -LASTLD(MEM),CENTY(MEM)=CENTY,ATTRIB(MEM)=ATTRIB)

(76)    INTSENT1    -->
                  SENT(%INTSENT1,-PRED,-SUCC,-CONDITN,QUESTSW(MEM),
                       LASTME(MEM)=@MOBENATR,LASTSE(MEM)=LOCOBJ(LOCATION),
                       LASTLD(MEM)=SUP(LOCATION),CENTY(MEM)=CENTY,
                       ATTRIB(MEM)=ATTRIB,ATTRIB.EQ.MOBENATR,-ATTRIB)

(77)    INTSENT2    -->
                  SENT(%INTSENT2,INTRGPH=" WHAT",
                       SUBJECT=@MOBENATR(AENTY(SEG))(AENTY),
                       QUESTSW(MEM),LASTME(MEM)=SUBJECT,
                       LASTSE(MEM)=LOCOBJ(LOCATION),LASTLD(MEM)=SUP(LOCATION))

(78)    INTSENT3    -->
                  SENT(%INTSENT3,SUPSET(MEM)='VALU',-@ATTRIB,
                       ATTRIB(MEM)=ATTRIB)
```

APPENDIX E

SUMMARY OF INTERROGATOR SEGMENT TYPES

INTERROGATOR--top level SEGMENT TYPE
- Attributes: None.
- Function: Initiates the INTERROGATOR rules. Creates SETAGL, RECLISTI, and MEMREC1 segments.

RECLISTI--created from INTERROGATOR
- Attributes: Those of the appropriate ACTNLIST, MOBLIST, or STALIST, plus LIST and LC. K is an attribute of the second copy of the ACTNLIST.
- Function: Makes available lists of the appropriate IPR records to be investigated by the INTERROGATOR. Creates a RECI segment followed by a RECLISTI with its LC attribute incremented by 1, or a NULL segment.

RECI--created from RECLISTI
- Attributes: Those of the IPR action record, mobile entity record, or stationary entity record of which it is a copy, plus K and LIST from the originating RECLISTI, and CENTY.
- Function: Creates the major working segments of the INTERROGATOR (ACTNREC1's, MOBREC1's, STAREC1's, and ACTNREC3's) or goes to NULL.

ACTNREC1--created from RECI, ACTNREC2, REC1, and SUCCERR
- Attributes: Those of the segment from which it was created, to include CENTY, ATTRIB, and possibly others.
- Function: Initiates the major investigations conducted by the INTERROGATOR. Inspects MOBENATR, LOCATION, IETM, DURATION, ASNDISTR, and SUCC attributes. Creates INTSENT1, QUANREC, ACTNREC2, INTSENT3, REC1, INTSENT2, SCSRREC1, and ICHECKED segments.

ACTNREC2--created from ACTNREC1
- Attributes: Those of the ACTNREC1 from which it was created but with one attribute removed.
- Function: Saves a copy of an ACTNREC1 to be tested further if the test initiated at the same time the ACTNREC2 was created is passed. Creates an ACTNREC1 or a NULL segment.

ACTNREC3--created from RECI
- Attributes: Those of the RECI from which it was created.
- Function: Conducts a check to insure that each action follows some other action.

MOBREC1--created from REC1

Attributes: Those of the REC1 from which it was created.

Function: To inspect mobile entity records in a manner comparable to ACTNREC1. Currently, the only attribute being investigated is WEIGHT. Creates a QUANREC or ICHECKED segment.

STAREC1--created from REC1

Attributes: Those of the REC1 from which it was created.

Function: To inspect stationary entity records in a manner comparable to ACTNREC1. Currently, no attributes are being investigated. Creates an ICHECKED segment.

MEMREC1--created from the INTERROGATOR

Attributes: None.

Function: Inspects PROBTIME(MEM) and TIMUNIT(MEM). Creates PHRASE or NULL segments.

ICHECKED--created from ACTNREC1, MOBREC1, and STAREC1

Attributes: Those of the segment from which it was created.

Function: Sets the CHECKED indicator of the original IPR record whose inspection has just been completed. Creates a NULL segment.

QUANREC--created from ACTNREC1, MOBREC1, DSTRREC1, and REC2

Attributes: Those of an attribute of the segment from which it was created, plus the CENTY, AENTY, ATTRIB, ATTRIB1, and VALSET attributes of the segment.

Function: Conducts checks on quantities to insure that the units are correct. Creates NULL, DSTRREC1, INTSENT1, INTSENT3, and REC1 segments.

DSTRREC1--created from QUANREC

Attributes: Those of the QUANREC from which it was created, to include CENTY, AENTY, ATTRIB, ATTRIB1, and VALSET. CENTY is changed to point to the distribution record in the IPR.

Function: Inspects the standard distributions available. Checks the mean and creates DSTRREC2 segments to continue the inspection. Also creates DSTRREC3, INTSENT3, and QUANREC segments.

DSTRREC2--created from DSTRREC1

Attributes: Those of the DSTRREC1 from which it was created.

Function: Inspects the standard deviation of normal distributions and the range of uniform distributions, or goes to NULL for exponential distributions. Creates DSTRREC3, INTSENT3, and QUANREC segments.

DSTRREC3--created from DSTRREC1 and DSTRREC2

Attributes: Those of the segment from which it was created, to include CENTY, AENTY, and ATTRIB.

Function: Produces a detailed description of the distribution except when in the usual question-answer mode. Creates INTSENT1 or NULL segments.

SCSRREC1--created from ACTNREC1 and SCSRREC1

Attributes: Those of the segment or attribute of a segment from which it was created, to include CENTY, AENTY, ATTRIB, and ATTRIB1.

Function: Investigates the various types of successors, to include QTYP, PTYP, FRACTNL, FTYP, STYP, and actions. Attributes inspected are OPENACT, CLOSACT, SUCARG, ACT2, and XYLAST. Creates SCSRREC1, SCSRREC2, SENT, PHRASE, SUCCERR, INTSENT2, ACTNREC1, and NULL segments.

SCSRREC2--created from SCSRREC1

Attributes: Those of the SCSRREC1 from which it was created, to include ATTRIB1 and AENTY.

Function: Operates in the same manner as DSTRREC3. Produces a description of the successor except when in the usual question-answer mode. Creates SUCCDESC or NULL segments.

REC1--created from ACTNREC1, QUANREC, SCSRREC1, and REC1

Attributes: Those of the segment or attribute of a segment from which it was created, to include CENTY, AENTY, and ATTRIB.

Function: Checks those records which represent functions to be sure all the points are there. Creates REC2, REC3, PHRASE, INTSENT1, INTSENT3, REC4, ACTNREC1, and NULL segments.

REC2--created from REC1 and REC2

Attributes: Those of the segment from which it was created, plus XYC.

Function: Inspects a record with a SUP of 'TYPTABL' to insure that its Y values have correct units. Creates QUANREC, REC2, and NULL segments.

REC3--created from REC1

Attributes: Those of the REC1 from which it was created, plus XYC=XYLAST-1

Function: Inspects the last X value of records with a SUP of 'TYPDIST' or 'FRACTNL' to insure that it is sufficiently close to 1000. Creates PHRASE, REC4, and NULL segments.

REC4--created from REC1 and REC3

Attributes: Those of the segment from which it was created.

Function: Produces an appropriate segment which in turn describes an erroneous assignment statement or successor. Creates ASNERR or SUCCERR segments.

ASNERR--created from REC4

Attributes: Those of the REC4 from which it was created.

Function: Produces a description of an erroneous assignment distribution. Creates ASNDESC and PHRASE segments.

SUCCERR--created from SCSRREC1 and REC4

Attributes: Those of the segment from which it was created, to include AENTY and CENTY.

Function: Produces a description of an erroneous successor. Creates SUCCDESC and ACTNREC1 segments.

INTSENT1--created from ACTNREC1, QUANREC, DSTRREC3, and REC1

Attributes: Those of the segment or the attribute of a segment from which it was created, to include CENTY, AENTY, ATTRIB, and usually MOBENATR and LOCATION.

Function: Produces an appropriate sentence or question and sets QUESTSW(MEM), CENTY(MEM), and ATTRIB(MEM). Creates a SENT segment.

INTSENT2--created from ACTNREC1, ACTNREC3, and SCSRREC1

Attributes: A SUP of 'DO', AENTY, CENTY, and ATTRIB from the segment, and one of PRED, SUCC, or CONDITN.

Function: Produces a question about what some mobile entity does before, after, or other than some other action. Creates a SENT segment.

INTSENT3--created from ACTNREC1, QUANREC, DSTRREC1, DSTRREC2, and REC1

Attributes: Those of the segment from which it was created, plus ATTRIB.

Function: Produces a basic question. Creates a SENT segment.

LIST OF REFERENCES

1. Minsky, Marvin, ed., Semantic Information Processing, MIT Press, Cambridge, Mass., 1968

2. Simmons, R. F., "Natural Language Question Answering Systems: 1969," Communications of the ACM, Vol. 13, No. 1, pp. 15-30, January 1970

3. Heidorn, George E., Natural Language Inputs to a Simulation Programming System--An Introduction, Naval Postgraduate School Technical Report No. NPS-55HD71121A, December 1971

4. Chomsky, Noam, Aspects of the Theory of Syntax, MIT Press, Cambridge, Mass., 1965

5. Lamb, Sydney M., Outline of Stratificational Grammar, Georgetown University Press, Washington, D. C., 1966

6. Hansen, Richard C., GES: A Data-Structure-to-GPSS Encoding System, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1970

7. McGee, Robert T., The Translation of Data Structure Representations of Simple Queuing Problems into GPSS Programs and English Text, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1971

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Asst Professor George E. Heidorn, Code 55Hd Department of Operations Research Naval Postgraduate School Monterey, California 93940	5
4.	Captain F. H. Hemphill, Jr., USMC Subunit 1, MCTSSA MCAS(H), Santa Ana, California 92710	1

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

REPORT TITLE
Computer Verification of the Completeness of a Simulation Problem Description by Natural Language Interaction

DESCRIPTIVE NOTES (Type of report and inclusive dates)
Master's Thesis; December 1971

AUTHOR(S) (First name, middle initial, last name)
Frederick Harold Hemphill, Jr.

REPORT DATE
December 1971

7a. TOTAL NO. OF PAGES
93

7b. NO. OF REFS
7

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT
Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

A research project in natural language man-machine communication is currently being conducted at the Naval Postgraduate School. The system being developed, called NLPQ, is an application of a more general system, known as NLP, which consists of a rule language and the programs to compile and execute those rules. NLPQ currently consists of particular sets of NLP rules which allow a user at a time-sharing terminal to input an English text description of a queuing problem, have the computer construct an internal problem representation, and then have it produce an English text description of the problem and a GPSS simulation program to solve the problem.

The research described in this thesis produced the INTERROGATOR, a set of NLP rules for inspecting the internal problem representation to insure that it is ready to produce a GPSS program. The INTERROGATOR produces questions about missing or erroneous information, which the user may then respond to in English. It may also be used in a question-answer mode to input the entire problem.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Natural Language						
Question-Answer System						
Simulation						
Computer						
Interaction						
Queuing						
GPSS						

Thesis
H43873 Hemphill
c.1
143220
Computer verification of the completeness of a simulation problem description by natural language interaction.